UNIVERSITY OF CALIFORNIA,
IRVINE

Deep Learning Models for Spatio-Temporal Forecasting and Analysis

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Reza Asadi

Dissertation Committee:
Professor Amelia Regan, Chair
Professor Michael Dillencourt
Professor R. Jayakrishnan

2020

DEDICATION

To my family for their love and support

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my deepest sincere gratitude to my advisor, professor Amelia Regan. Her guidance, support, optimism and encouragement have been invaluable throughout the entire time of my PhD studies. I am grateful to the committee members of my final defense, Professor Michael Dillencourt and Professor Jay Jayakrishnan. I am also grateful to professor Solmaz Kia, for our collaborations in part of my PhD studies. Her guidance and expertise were instrumental in guiding my early research projects. I would like to express my special thanks to my master degree advisor, Professor Mehdi Ghatee, whose encouragement, wisdom and expertise was deeply motivating and inspiring for my academic pursuits.

Also, I'd like to say thanks to my colleagues during my entire PhD studies. The discussions with Dmitri Akhripov were immensely formative and influential early on. I am fortunate to have the support of great friends in department of computer science, Ahmad Razavi, Siavash Rezaei, Saeed Mirza Mohammadi and Mehdi Torabzadeh. I am also glad to have present members of Amelia Regan's lab, Amari Lewis, Arash Nabili, Dalal Alharthi, and Caesar Aguma. I also learned a lot from my collaborators, whom I have crossed paths during my experience in industry, Rodrick Megraw, Xiaoxia Shi and Kelsey Dilullo. Throughout the entire of my PhD, I've had valuable experience as a teaching assistant of graduate classes, and I've learned valuable skills from Professors Rick Lathrop, Michael Dillencourt and Pierre Baldi.

I thank Institute of Electrical and Electronics Engineers (IEEE), Elsevier, Association for Computing Machinery (ACM) for giving me permissions to include my previously published papers in this dissertation.

VITA

Reza Asadi

EDUCATION

Doctor of Philosophy in Computer Science **2020**
University of California, Irvine *Irvine, California*

Master of Science in Computer Science **2016**
University of California, Irvine *Irvine, California*

Master of Science in Computer Science **2013**
Amirkabir University of Technology *Tehran, Iran*

Bachelor of Science in Computer Science **2011**
Amirkabir University of Technology *Tehran, Iran*

RESEARCH EXPERIENCE

Graduate Research Assistant **2014–2019**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Introduction to Artificial Intelligence, Professor Rick Lathrop **Winter, 2018**
University of California, Irvine

Advanced Data Structures, Professor Michael Dillencourt **Spring, 2018**
University of California, Irvine

Introduction to Artificial Intelligence, Professor Kalev Kask **Fall, 2018**
University of California, Irvine

Deep Learning and Neural Networks, Professor Pierre Baldi **Winter, 2019**
University of California, Irvine

REFEREED JOURNAL PUBLICATIONS

A Spatio-Temporal Decomposition Based Deep Neural Network for Time Series Forecasting **2019**
Reza Asadi and Amelia Regan; Journal of Applied Soft Computing - Elsevier

Cycle flow formulation of optimal network flow problems and respective distributed solutions **2019**
Reza Asadi and Solmaz S. Kia; IEEE/CAA Journal of Automatica Sinica

A Rule-Based Decision Support System in Intelligent Hazmat Transportation System **2015**
Reza Asadi and Mehdi Ghatee; IEEE Transaction on Intelligent Transportation Systems

REFEREED CONFERENCE PUBLICATIONS

Spatio-Temporal Clustering of Traffic Data with Deep Embedded Clustering (Best Paper Award) **Nov 2019**
Reza Asadi and Amelia Regan; Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Prediction of Human Mobility

A Convolutional Recurrent Autoencoder for Spatio-temporal Missing Data Imputation **July 2019**
Reza Asadi and Amelia Regan; International Conference of Artificial Intelligence, ICAI'19

Cycle basis distributed ADMM solution for optimal network flow problem over biconnected graphs **July 2016**
Reza Asadi, Solmaz S. Kia and Amelia Regan; 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)

ABSTRACT OF THE DISSERTATION

Deep Learning Models for Spatio-Temporal Forecasting and Analysis

By

Reza Asadi

Doctor of Philosophy in Computer Science

University of California, Irvine, 2020

Professor Amelia Regan, Chair

Spatio-temporal problems arise in broad areas of environmental and transportation systems. These problems are challenging, because of both spatial and temporal neighborhood similarities and correlations. We consider traffic data, which is a complex example of spatio-temporal data. Traffic data is geo-referenced time series data, where fixed locations have observations for a period of time. Traffic data analysis and related machine learning tasks have an important role in intelligent transportation systems, such as designing navigation systems, traffic management, control systems and in the future will be essential for setting appropriate anticipatory tolls. Recent data collection methodologies dramatically increase the volume of available spatio-temporal data, which require scalable machine learning models. Moreover, deep learning models outperform traditional machine learning and statistical models due to their strong feature learning abilities in spatial and temporal domains. Increases in available data and recent advances in deep learning models in spatio-temporal domains are the main motivations of this dissertation.

We first study, non data-driven and optimization-based solutions for the network flow problem, which appears in a wide range of applications including transportation systems and electricity networks. In these applications, the underlying physical layer of the systems can generate a very large graph resulting in an optimization problem with a large decision vari-

able space. We present a distributed solution for the network flow problem. The model uses cycle basis and an alternating direction method of multipliers (ADMM) method to find a lower computational time and number of communications, while obtaining a centrally optimal solution.

Second, we attempt to obtain spatio-temporal clusters in traffic data, which represent similar traffic data in terms of both spatial and temporal similarities. Clustering of traffic data are used to analyze traffic congestion propagation and detection. We obtain spatio-temporal clusters using a modification to Deep Embedded Clustering, which considers both spatial and temporal similarities in latent features. Also we define new evaluation metrics to evaluate spatio-temporal clusters of traffic flow data.

Third, when sensors collect spatio-temporal data in a large geographical area, the existence of missing data cannot be escaped, which negatively impacts of prediction models. Here, we investigate the problem of incorporating both spatial and temporal contexts in missing traffic data imputation using convolutional and recurrent neural networks. We propose a convolutional-recurrent autoencoder for missing data imputation, and illustrate the performance of autoencoders for missing data imputation in spatio-temporal data.

Finally, traffic flow prediction has an important role in diverse intelligent transportation systems and navigational systems. There is a large literature on this problem. However, the problem is challenging for high-dimensional traffic data. We explicitly design the neural network architecture for capturing various types of spatial and temporal patterns. We also define evaluation metrics for spatio-temporal forecasting problems to better evaluate generalization of the model over various spatial and temporal features.

Chapter 1

Introduction

Here, we describe main characteristics of spatio-temporal traffic data, along with the machine learning tasks related to traffic data.

1.1 Spatio-temporal traffic data

Spatio-temporal data have both contexts of space and time. In a spatial area, sensors observes an object state, event, or position over a time period. Spatio-temporal problems can be classified into five categories, events, Geo-referenced data, Geo-referenced time series, moving objects and trajectories [8]. We consider Geo-referenced time series data, which represent the history of observed variables over a time period in fixed locations of a geographical area. Spatio-temporal data mining arise in various domains [132], including transportation science (traffic flow prediction and analysis [131], [76], [144]), environmental science (participation, weather and wind forecasting [121], [113]), load forecasting[120], and Social sciences [22]. Spatio-temporal data are multi-variate time series data, where there are spatial similarities and correlations between neighboring time series. There are multiple reasons that

make spatio-temporal data complex and challenging. First, such a data have both spatial and temporal contexts at the same time. Most of the datasets have only one of the contexts. For example, generally images are an example of spatial data. Also, most of text processing problems include temporal, or sequential, relation among the words and characters. Considering both of the contexts require careful design of the machine learning problems. Second, spatio-temporal data are gathered over a network or geographical area with sensors. Such a data can be represented with graph-structured data. Learning a graph-structured data necessitate careful design of the machine learning models. Third, spatio-temporal features are not randomly independent. There are non-linear similarities and correlations in both spatial and temporal contexts. It makes the data more difficult to work with than general randomly and independent distribution. Lastly, there are various machine learning tasks related to spatio-temporal data, which require careful design of the machine learning model, such as missing data imputation, forecasting, clustering and anomaly detection.

In this dissertation, we consider Geo-referenced data, where specific locations have their own time series data. Also, time series data are gathered in synchronous way. We describe a spatio-temporal data with $\mathbf{X} \in \mathbb{R}^{s,\bar{t},f}$, where s is the number of spatial points, also known as sensors, $\bar{t}$ is the total number of time stamps, and f is the number of features. Sensors gather such data in a geographical area, so it is considered as the first dimension. Given a large value of $\bar{t}$, a sliding window method with window size of w generates a sequence of data points $\mathbf{x}^t \in \mathbb{R}^{s,w,f}$ for $t \in \{1, \ldots, \bar{t}\}$. The machine learning models receive input data points, individually $\mathbf{x}_i^t \in \mathbb{R}^{w,f}$ for one location i or globally $\mathbf{x}^t \in \mathbb{R}^{s,w,f}$. A sliding window method increase the flexibility of designing machine learning models that are capable of processing input data over different temporal or spatial features.

Traffic data are obtained with loop-detectors, cameras, and other types of sensors gather traffic data. The data represents traffic flow, speed and occupancy. We use traffic flow data from the Bay Area of California represented in Fig. 5.3 which have been broadly used,

and available in PeMS [1]. The traffic data is gathered every 30 seconds and aggregated over 5 minute periods. Each sensor on highways has flow, occupancy and speed at each time stamp. A sensor is an inductive loop traffic detector device on mainline, off-ramp or on-ramps locations.

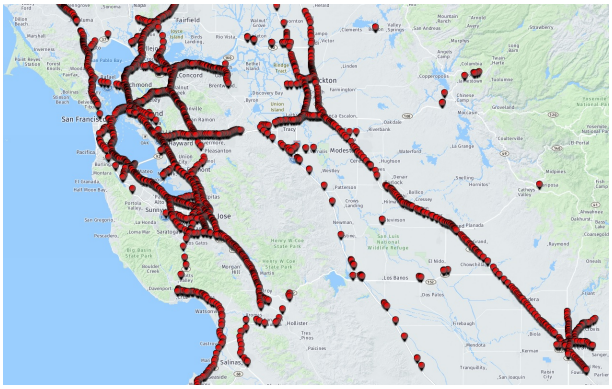
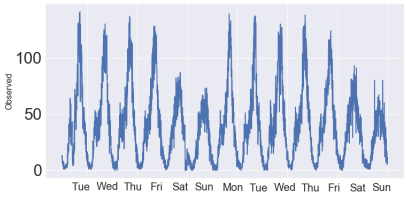


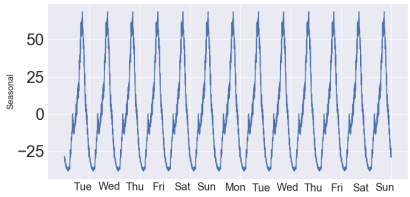
Figure 1.1: Representation of loop detector sensors on highways of Bay Area, California

Spatial patterns in traffic data are the results of traffic evolution in the network. Here, we analyze the spatial, short-term and long-term patterns. In Fig. 5.4, an additive time series decomposition of traffic flow data is presented for one station. Given a one day frequency, time series decomposition has similar, repeated (seasonal) daily patterns. Moreover, there are long-term weekly patterns, shown as trends $\mathbf{T}$. The long-term patterns, such as seasonal and trends, arise from similar periodic patterns, generated outside of the highway network. In other words, they are related to origin-destination matrix of vehicles in the network. The time series residuals are not only random noise, but also the results of spatial and short-term patterns, related to the evolution of traffic flow or sudden changes in smaller regions of the network.

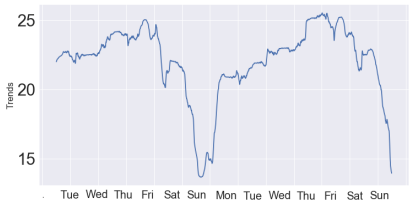
Time series residuals are interpreted as random noise for time series data. However, in traffic flow data, the residuals are the results of traffic evolution in the network. In Fig. 5.5, we examine the non-linear relation of flow, speed and occupancy in one day and one sensor. It shows that high occupancy reduces speed in a road segment, which is the result of traffic congestion. For more details, we refer the reader to the theoretical analysis of



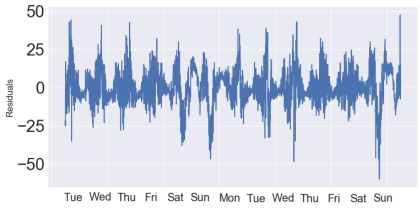
(a) The observed traffic flow data.



(b) Seasonality of traffic flow data



(c) Trends of traffic flow data

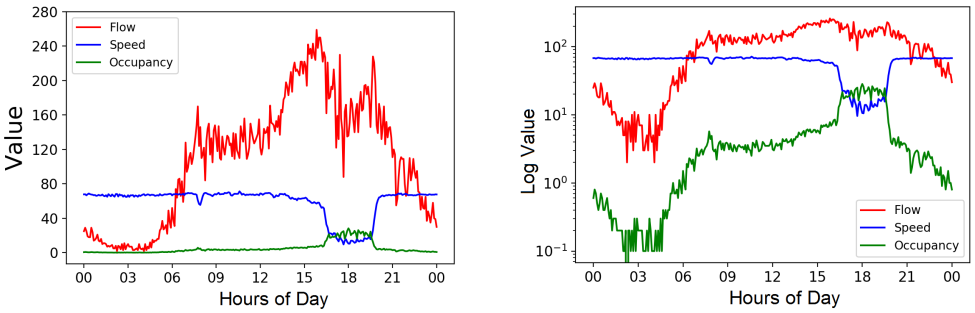


(d) Residuals of traffic flow data

Figure 1.2: Time series decomposition with daily frequency is represented for one sensor's traffic flow data.

these relationships in [38] and [89], respectively. In a transportation network, the congestion propagation describes the relation among neighboring sensors of a highway, shown in Fig. 5.6. For a given three sensors, traffic congestion is propagated with nearly 20 minutes of delay. For a larger geographical area, the speed of 13 successive sensors is represented in an image-like representation of spatio-temporal data, in Fig. 5.7. The reduction of speed in peak hours is presented with darker colors. It shows that the reduction in speed is similar in neighboring areas, which also represents the existence of spatial correlation in neighboring sensors.

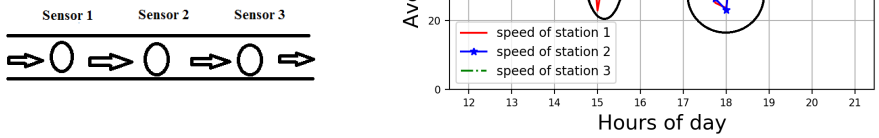
One of the methods of finding spatial similarities in spatio-temporal data is to find the similarity matrix among locations. First, we need a distance function among time series. Dynamic Time Warping as a distance function of time series [] is described in Chapter 4. Here, to represent the spatial similarities, a similarity matrix is obtained for a given 25 sensors in a highway. The result is represented with a heatmap. We can see that the elements close to diagonal have lower values. Also, a clustering method can find the clusters of similar sensors.



(a) The relation between flow, occupancy and speed is shown. Occupancy, with value more than 8% occupied by vehicles, decreases average speed, which is the result of traffic congestion.

(b) Log plot to represent the linear relation between occupancy and flow in free flow speed (about 70 mph).

Figure 1.3: The relation between occupancy, speed and flow



(a) Three successive sensors are selected to represent congestion propagation in the network.

(b) The reduction in speed of sensor 1 and 2 can be observed twice in this plot, in which there is 20 minute delay due to congestion propagation delay time.

Figure 1.4: The congestion propagation in successive sensors.

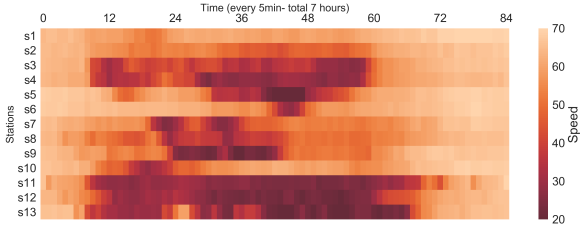


Figure 1.5: Image-like representation of a speed value of 13 successive sensors over 7 hours (5 min time stamp)

These analysis show that spatial and temporal patterns of traffic data have some unique characteristics. Here, we address some of the important challenges in designing a neural

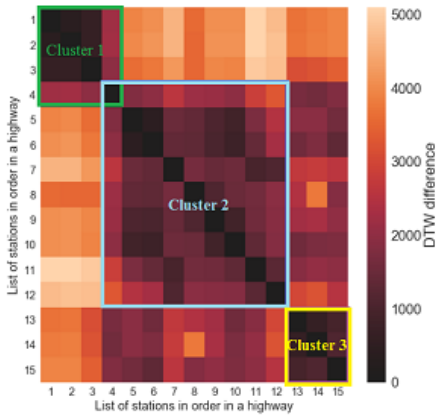


Figure 1.6: A DTW distance of pair sensors are illustrated. A Clustering method finds three clusters. Such a plot show the spatial similarity in the traffic flow data.

network architecture for spatio-temporal data. In these data, time series residuals are not meaningless noise, but they represent spatial patterns in data. Moreover, convolutional neural networks capture spatial patterns. However, convolutional layers can capture spatial and short-term patterns, but sliding a convolutional kernel on spatial features miss the underlying structure of the network and reduce the performance of the model. Also, spatio-temporal data have a non-stationary behaviour in the existence of long-term patterns. Furthermore, spatial and temporal patterns can be used to handle missing data.

1.2 Machine learning problems in traffic data

We describe the main characteristics of spatio-temporal data, and illustrate it on traffic data in last section. Here, we describe the importance of main machine learning problems arise for traffic data and machine learning solutions for such problems.

1.2.1 Traffic flow prediction

Short-term traffic flow prediction is an important tool for reducing travel time, and helps traffic managers to understand traffic dynamics, and reduce traffic congestion, transportation costs and air pollution [21], [112]. Traffic data include complex spatial and temporal patterns, but it is not chaotic and prediction models can anticipate near future traffic states [25]. Navigational systems on smart phones and Map software applications have an important role in our daily lives, short-term traffic flow prediction is a component in predicting travel time. There is a large number of research studies for the traffic flow prediction. Starting in the 1970's with the original work of Gazis and Knapp [45], many studies develop new models for traffic flow prediction problems, such as auto-regressive integrated moving average (ARIMA) [57] and Seasonal-ARIMA [68], and statistical techniques, such as Markov chain models [140] and Bayesian networks [129]. However, there are several limitations on these models, due to prior assumptions, lack of handling missing data, noisy data, outliers, as well as the curse of dimensionality. Recently, deep learning models have been successfully applied to the traffic flow prediction [80], [134], [100].

1.2.2 Imputing incomplete traffic data

Traffic data are obtained by a large number of sensors. A challenge in data processing step is to have missing values. This is the result of malfunctioning of hardware devices, communication network problems, power supply issues, expected or unexpected maintenance time and so on [20]. While various solutions reduce the size of missing values, this ideal solutions rarely occur. Hence, researchers study missing data imputation in broad areas, including transportation science. There are a large number of studies develop machine learning solutions for missing data imputation of traffic data. Existence of spatial and temporal neighborhood similarity makes the problem challenging. Simulation-based models have been proposed for

missing data imputation [90]. In [77], a k-nearest-neighbor models impute incomplete traffic data based on similarity of spatial and temporal features. Low-dimensional representation and learning of spatio-temporal data has been increasingly popular in missing data imputation models. In [104] and [103], a Bayesian-PCA and probabilistic-PCA are proposed to impute the incomplete traffic flow volume data. Clustering based solutions [67], tensor-based completion [67], and spatio-temporal cokriging [67] are other types of machine learning models that have been studying missing data imputation problem for traffic data. However, when there are big data sets, scalable machine learning models, such as deep learning models require careful design to consider missing data imputation for spatio-temporal traffic data [43], [145].

1.2.3 Clustering of traffic flow data

Clustering of traffic data is an important tool for analysis of traffic data. In [39] and [18], they propose a model for classifying traffic data into similar groups using fuzzy and density-based clustering models. Also, clustering models have been used for spatio-temporal detecting congestion patterns [7], [106]. Moreover, clustering of traffic data can be used as a component for other applications. In [67], they propose a clustering-based model for missing data imputation of traffic data. In [93], they propose a modified k-means clustering for the travel time prediction. In [79], they propose a clustering-based model for detecting anomalies in traffic data. Broad range of applications for using clustering as a method of analysis or improving the performance of other machine learning models, show the importance of proposing a clustering model using deep learning.

1.3 Outline and contributions

The structure of the dissertation and contributions of each project is as follows,

Chapter 2 describe a new solution for static network flow problem with capacity bounds. The objective of this project is to find a distributed solution for network flow problem and reduce communication and computation costs. We consider static network flow problem with an optimization solution, contrary to other chapters, where data-driven solutions are studied. We investigate the graph and optimization theoretical concepts in finding a cycle basis solution and an efficient distributed solution.

Contributions of Chapter 2 include:

- We take advantage of the cycle basis concept in the graph theory to reduce the search variables of an optimal network flow.
- We apply the cycle-basis solution to power flow control with generators and storage.
- We propose an ADMM solution for a distributed cycle-basis network flow problem.
- Numerical example demonstrates the outperformance of the proposed model in reducing computational time and communications.

Chapter 3 describe a clustering model for spatio-temporal traffic data. The focus is to find spatial and temporal clusters with deep learning models. Both types of spatial and temporal clusters are obtained and evaluated on traffic data.

Contributions of Chapter 3 include:

- We describe the advantages of using deep embedded clustering for clustering of spatio-temporal traffic data.

- We propose a modified deep embedded clustering, which uses prior geographical information to find spatial clusters.

Chapter 4 describe missing data imputation problem for spatio-temporal data. The focus is to design a convolutional-recurrent autoencoder for imputing missing values. The evaluation shows that the imputation of artificial missing values is better than state-of-the-art neural network models.

Contributions of Chapter 4 include:

- We propose a convolutional bidirectional-LSTM for capturing spatial and temporal patterns.
- we analyze an autoencoder's latent feature representation in spatio-temporal data and illustrate its performance for missing data imputation.
- The result shows that multiple missing data imputation with reconstruction of input can outperform baseline and state-of-the-art neural networks.

Chapter 5 describe traffic flow prediction problem. The focus is to decompose spatial and temporal features and use them in designing a deep learning model for spatio-temporal forecasting problem. There is a rigorous analysis of spatio-temporal features and evaluation of the models.

Contributions of Chapter 5 include:

- A deep neural network is proposed for the short-term spatio-temporal forecasting.
- A clustering method and a multi-kernel convolutional layer capture spatial patterns.
- Time series decomposition helps the model to better capture temporal patterns.

- The model generates more robust outcomes when faced with missing data.
- The performance of the model is evaluated for spatio-temporal forecasting problem using new evaluation metrics.

Chapter 2

Distributed network flow problem

Minimum cost network flow problems appear in a wide range of applications including general network optimization problems, transportation systems, electricity networks and information networks. In these applications, the underlying physical layer of the systems can generate a very large graph resulting in an optimization problem with a large decision variable space. Various arc and node based cyber layer layouts have been proposed to solve this problem in a distributed manner. In this chapter, for a physical layer network of n nodes and m arcs with biconnected graph topology, we take advantage of the cycle basis concept in the graph theory to reduce the search variables of an optimal network flow from m to $m-n+1$ variables. We show that our proposed new formulation of the optimal network flow problem is amenable to a distributed solution using alternating direction method of multipliers (ADMM) method via a cyber layer whose nodes are defined based on the fundamental cycles of a cycle basis of the physical layer graph. We demonstrate the performance of our algorithm through a numerical example. The results of this work is published in [12] and the extension of the work is published in [11].

2.1 Introduction

In this chapter, we consider the optimal network flow problem in a network of a physical system. The physical system consists of several routes between a source point and a sink point, which are used to transfer a flow from the source to the sink. The objective of the optimum network flow problem is to minimize the overall cost of transporting flow [23]. Network flow problems appear in many important applications, such as power networks [92], communication networks [122], wireless sensor networks [133] and transportation systems [19]. Network flow problems and their variants are also relevant to computer vision [64], robust routing with the objective of routing with minimal variance in noisy communications [133], wireless routing and resource allocation [135]. With the advent of new technologies, the amount of available data and network size has been increasing, which necessitate various performance improvement techniques in cyber-physical systems [107], and increase the size of optimization problems. However, the number of decision variables is directly related with the time and space (resources1) computation complexity of optimization solvers.

Optimal network flow problems are normally cast as a convex optimization problem where the cost is the sum of convex cost of flows through arcs subject to capacity bounds for each arc and flow conservation equations at each node. With the advent of new technologies, both the size of networks and the amount of data available on those networks have been increasing. Such expansions in the size of physical networks result in increasingly large optimization problems associated with optimal network flow problems. This has promoted the interest in the parallel and distributed solutions for optimal network flow problems [59], [98].

Convex cost network flow problems can be solved in a distributed manner via dual sub-gradient descent [23]. Sub-gradient methods are analyzed for distributed convex optimization [94] and uncertainty of the network structure is considered in [78]. The resulting algorithm has a slow convergence rate [114]. In [142], they proposed a second order method

for network flow optimization, which has better convergence rate of sub-gradient methods. In [88], they applied local domain ADMM on minimum cost flow problem.

However, all of the above algorithms assume arc based network flow, that is, the total number of search variables matches the total number of arcs in the physical network. To solve the network flow problem in a distributed manner, each arc or group of arcs are assigned to a cyber layer node. Then, the optimization problem of the optimal network flow problem is cast in a separable manner and solved by cyber layer nodes in a cooperative way. Although, in distributed algorithms the computational cost of the optimal flow problem is distributed among the agents in cyber layer, the large number of decision variables normally translates to a large number of cyber nodes or large communication overhead between neighboring cyber nodes.

In this chapter, we use the cycle basis concept in graph theory to reduce the search variables of an optimal network flow over a physical layer graph of n nodes and m arcs, with biconnected topology, from m variables to $m - n + 1$. The concept of cycle basis for network flow optimization is related to the tie-set graph theory to solve current and voltage Kirchhoff laws in electric circuits [63]. Using the tie-set graph theory, [62] and [92] have discussed a method to solve optimal network flow problems with quadratic arc costs functions.

In this chapter, we consider a minimum network flow problem where the arc costs are convex functions. Our contributions in this chapter are a rigorous study of the relationship between the fundamental cycle basis of a biconnected graph and the oriented incidence matrix of the connected physical layer graph of a network flow problem. Recall that the incidence matrix can be used to cast the flow conservation requirement at each node of the network flow problem. Here, we show that the nullspace of the incidence matrix is cast by rows of a cycle basis matrix of the graph, resulting in a systematic way to eliminate the affine flow conservation constraints and replacing the arc flow by an equation that is described based on the cycle basis matrix, cycle flow and a particular solution of the flow conservation

equation. Our next contributions are (a) results that show how particular solution needed in the reduced variable representation can be found in an efficient manner; (b) results that show the new formulation of the optimal network flow problem based on the cycle basis variables is amenable to a distributed ADMM solution method (c.f. [26] and [88]) via a cyber layer whose nodes are defined based on the fundamental cycles of a cycle basis of the physical layer graph. We also discuss other possible cyber layer choices that are still able to solve our formulation of the optimal network flow problem in a distributed manner.

The chapter is organized as follows. Section 2.2 defines our notation, graph theoretic terminologies and concepts, and reviews some preliminary results from the literature on convex optimization. Section 2.3 formally presents our problem statement. Section 2.4 gives our main results on cycle basis formulation of the optimal network flow problem and discuss how this formulation can be solved in a distributed manner via a distributed ADMM algorithm. Section 2.5 demonstrates the performance of our algorithm using a numerical example. Section 2.6 gives the summary and states the future work.

2.2 Preliminaries

In this section, we introduce our notation, briefly review the relevant graph theoretic definitions and concepts, as well as, some relevant optimization theory results.

2.2.1 Notation

Let $\mathbb{R}$ and $\mathbb{R}_{>0}$ be, respectively, the set of real and positive real numbers. For a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ we represent its entry in i^{th} row and j^{th} column by $\mathbf{A}_{ij}$. Transpose of $\mathbf{A}$ is denoted by $\mathbf{A}^\top$. We let $\mathbf{0}_n$ denote the vector of n zeros. When clear from the context, we do not specify the matrix dimensions. In a network of n agents, if $\mathbf{p}_i \in \mathbb{R}^d$ is a variable of agent

$i \in \{1, \ldots, n\}$, the aggregated $\mathbf{p}_i$'s of the network is the vector $\mathbf{p} = [\mathbf{p}_1{}^\top, \cdots, \mathbf{p}_N{}^\top]^\top \in (\mathbb{R}^d)^n$. For finite sets V_1 and V_2, $V_1 \backslash V_2$ is the set of elements in V_1, but not in V_2. The cardinality of a finite set V is $|V|$.

2.2.2 Graph theory

In this section, following [41], we review our graph related terminology and conventions. We also introduce graph related notation that we use throughout the chapter.

We represent a graph with a set of nodes $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ and a set of arcs $\mathcal{E} = \{e_1, \cdots, e_m\} \in \mathcal{V} \times \mathcal{V}$ with $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The graph is assumed to be undirected and has no self-loops. The (unoriented) incidence matrix $\mathfrak{I} \in \mathbb{R}^{n \times m}$ of $\mathcal{G}$ is a matrix whose n rows correspond to the n nodes and the m columns correspond to m arcs such that $\mathfrak{I}_{ij} = 1$ of j^{th} arc is incident on the i^{th} node, and $\mathfrak{I}_{ij} = 0$ otherwise. A *walk* is an alternating sequence of nodes and connecting arcs. A *path* is a walk that does not include any node twice, except that its first node might be the same as its last. A graph is *connected* if there is a path from its every node to every other node. A graph is *biconnected* if and only if any node is deleted, the graph remains connected.

Next, we review some concepts and proprieties pertinent to cycles in graphs. A *cycle* of $\mathcal{G}$ is any subgraph in which each node has even degree. The degree of a node in a graph is the total number of arcs connected to that node. A *simple cycle* is a path that begins and ends on the same node with no other repetitions of nodes. For a graph with arc set $\mathcal{E} = \{e_1, \cdots, e_m\}$, a cycle vector $\mathbf{c} \in \mathbb{R}^m$ is a binary vector with $\mathbf{c}_i = 1$ if e_i is in the cycle and $\mathbf{c}_i = 0$, otherwise. The set of all cycle vectors on $\mathcal{G}$ is a linear vector space over a field, Galois field modulo 2 or $GF(2)$. In this field elements belong to the set $\{0, 1\}$ with operation addition modulo 2 written as '$\oplus$' such that $0 \oplus 0 = 0$, $1 \oplus 0 = 1$, $0 \oplus 1 = 1$, $1 \oplus 1 = 0$. Let graph $\mathcal{G}$ has μ cycles and m arcs. A cycle matrix $\mathbf{B} \in \mathbb{R}^{\mu \times m}$ of $\mathcal{G}$ is a binary matrix, where

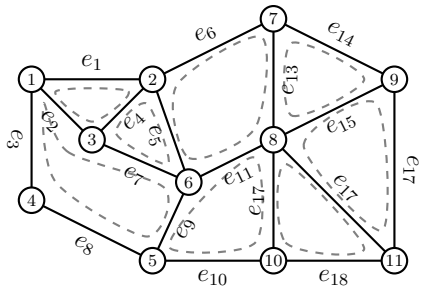


Figure 2.1: A biconnected graph of 11 nodes and 18 arcs with its cycle basis of dimension 8.

$\mathbf{B}_{ij} = 1$ if the j^{th} arc is in the i^{th} cycle of the graph, $\mathbf{B}_{ij} = 0$, otherwise, i.e., i^{th} row of $\mathbf{B}$ is the cycle vector of cycle $i \in \{1, \ldots, \mu\}$.

The following result gives the rank of the cycle matrix with in $GF(2)$.

Lemma 2.2.1 ($GF(2)$ rank of cycle matrix of a connected graph [41])**.** *If* $\mathbf{B}$ *is a cycle matrix of a connected graph* $\mathcal{G}$ *with* n *nodes and* m *arcs, then* $GF(2)$ *rank of* $\mathbf{B}$ *is* $\mu = m - n + 1$.$\square$

A *cycle basis* of $\mathcal{G}$ is a set of simple cycles that forms a basis of the cycle space of $\mathcal{G}$. Every cycle in a given cycle basis is called a *fundamental cycle.* $\mathbf{B}^{\text{f}} \in \mathbb{R}^{\mu^{\text{f}} \times m}$, where $\mu^{\text{f}} = m-n+1$ is a *fundamental cycle matrix* of graph $\mathcal{G}$ if and only if its rows span cycle space of $\mathcal{G}$ in $GF(2)$. Therefore, we have $\text{rank}(\mathbf{B}^{\text{f}}) = m - n - 1$ in $GF(2)$. A fundamental cycle basis of a graph is constructed by its spanning tree, in a way that cycles formed by a combination of a path in the tree and a single arc outside of the tree. For every arc outside of the tree, there exists one cycle. Each cycle generated in this way is independent of other cycles, because it has one arc, not exist in other cycles. Figure 2.1 depicts a graph with its cycle basis highlighted by the dashed closed curves.

In this chapter, we focus on *planar graphs.* A planar graph is a graph which can be drawn in the plane without any edges crossing. Some pictures of a planar graph might have crossing edges, but it must be possible to redraw the picture to eliminate the crossings.

Next, we discuss our conventions for the oriented form of a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. When

there is an orientation assigned to arcs of a graph, we write $e_k = (v_i, v_j)$ if arc e_k points from node v_i towards node v_j. Notice that in the oriented graph $\mathcal{G}^{\mathrm{o}} = (\mathcal{V}, \mathcal{E}^{\mathrm{o}})$, if $(v_i, v_j) \in \mathcal{E}^{\mathrm{o}}$ then $(v_j, v_i) \notin \mathcal{E}^{\mathrm{o}}$, i.e., there is no symmetric arc in the oriented graph. For an oriented graph $\mathcal{G}^{\mathrm{o}}$, the *oriented incidence* matrix is the matrix $\mathfrak{I}^{\mathrm{o}} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$, where $\mathfrak{I}^{\mathrm{o}}_{ij} = 1$ if arc e_j leaves node v_i, $\mathfrak{I}^{\mathrm{o}}_{ij} = -1$ if arc e_j enters node v_i, otherwise $\mathfrak{I}^{\mathrm{o}}_{ij} = 0$. For a connected graph of n nodes with a given orientation, the rank of both $\mathfrak{I}$ and $\mathfrak{I}^{\mathrm{o}}$ is $n-1$. For cycles in the oriented version of a graph of n nodes and m arc, $\mathcal{G}^{\mathrm{o}} = (\mathcal{V}, \mathcal{E}^{\mathrm{o}})$, we assign the clockwise direction as positive cycles orientation and define the *oriented cycle vector* $\mathbf{c}^{\mathrm{o}} \in \mathbb{R}^m$ with $\mathbf{c}^{\mathrm{o}}_i = 1$ if e_i is in the cycle and aligned with its direction, $\mathbf{c}^{\mathrm{o}}_i = -1$ if e_i is in the cycle but opposing the direction of the cycle and finally $\mathbf{c}^{\mathrm{o}}_i = 0$ if e_i is not in the cycle. In the following, we show that every oriented cycle matrix is in the nullspace of oriented incident matrix $\mathfrak{I}^{\mathrm{o}}$.

Theorem 2.2.1 (relationship between the oriented incidence and oriented cycle vector [41])**.** *Let $\mathcal{G}^o$ be an oriented graph with oriented incidence matrix $\mathfrak{I}^o$. Then, every oriented cycle vector $\mathbf{c}^o$ is orthogonal to every row of $\mathfrak{I}^o$, i.e., $\mathfrak{I}^o\,\mathbf{c}^o = \mathbf{0}_n$.* □

For a graph $\mathcal{G}$ of n nodes and m arcs with fundamental cycle matrix $\mathbf{B}^{\mathrm{f}} \in \mathbb{R}^{(m-n+1)\times m}$, when we assign orientation, we represent the *oriented fundamental matrix* by $\mathbf{B}^{\mathrm{of}}$. Next, we show that $\mathrm{rank}(\mathbf{B}^{\mathrm{of}}) = m - n + 1$ in real space.

Theorem 2.2.2 (rank of oriented cycle matrix of an orineted graph)**.** *Let $\mathbf{B}^{\mathrm{of}}$ be an oriented fundamental cycle matrix of an oriented graph $\mathcal{G}^o$ with n nodes and m arcs, then* $\mathrm{rank}(\mathbf{B}^{\mathrm{of}}) = m - n + 1$.

Proof. The proof is a straightforward consequence of how fundamental cycle basis is constructed from spanning tree of a graph. By construction, every fundamental cycle of a cycle basis has an arc that does not appear in other fundamental cycles. As a result, (in both unoriented and oriented cases) the cycle vector of each fundamental cycle is independent of all other fundamental cycle vectors in a cycle basis. This completes the proof. □

We close this section by introducing some notations for oriented fundamental cycles of an oriented connected graph $\mathcal{G}^{\mathrm{o}}$ with n nodes and m arcs. For a given cycle basis of $\mathcal{G}^{\mathrm{o}}$, we represent the set of its fundamental (oriented) cycles by $\mathcal{C}^{\mathrm{f}}$ and its dimension by $\mu^{\mathrm{f}} = m-n+1$. We represent the set of arcs of an oriented fundamental cycle $\mathfrak{C}_i^{\mathrm{of}} \in \mathcal{C}^{\mathrm{f}}$, $i \in \{1,\dots,\mu^{\mathrm{f}}\}$, by $\mathcal{C}_i^{\mathrm{ofe}} = \{e_j \in \mathcal{E}^{\mathrm{o}}, j \in \{1,\dots,m\} \,|\, \mathbf{B}_{ij}^{\mathrm{of}} \neq 0\}$. For a given cycle basis, we refer to the cycles that share an arc as neighbors and represent the set of (cycle) neighbors of any fundamental cycle $\mathfrak{C}_i^{\mathrm{of}} \in \mathcal{C}^{\mathrm{f}}$, $i \in \{1,\dots,\mu^{\mathrm{f}}\}$, by $\mathcal{N}_i^{\mathfrak{C}} = \{j \in \{1,\dots,\mu^{\mathrm{f}}\}\backslash\{i\} \,|\, \exists\, k \in \{1,\dots,m\} \text{ s.t. } \mathbf{B}_{ik}^{\mathrm{of}} \neq 0 \text{ and } \mathbf{B}_{jk}^{\mathrm{of}} \neq 0\}$. For every fundamental cycle $\mathfrak{C}_i^{\mathrm{of}}$, $i \in \{1,\dots,\mu^{\mathrm{f}}\}$ in a given cycle basis $\mathcal{C}^{\mathrm{f}}$ of a given $\mathcal{G}^{\mathrm{o}}$, we write its member arc set as $\mathcal{C}_i^{\mathrm{ofe}} = \bar{\mathcal{C}}_i^{\mathrm{ofe}} \cup \tilde{\mathcal{C}}_i^{\mathrm{ofe}}$, where $\bar{\mathcal{C}}_i^{\mathrm{ofe}} = \{e_j \in \mathcal{C}_i^{\mathrm{ofe}} \,|\, \mathbf{B}_{ij}^{\mathrm{of}} \neq 0 \text{ and } \mathbf{B}_{kj}^{\mathrm{of}} = 0,\ k \in \{1,\dots,\mu^{\mathrm{f}}\}\backslash\{i\}\}$ is the set of arcs that are only in $\mathfrak{C}_i^{\mathrm{of}}$ and $\tilde{\mathcal{C}}_i^{\mathrm{ofe}} = \mathcal{C}_i^{\mathrm{ofe}}\backslash\bar{\mathcal{C}}_i^{\mathrm{ofe}}$ is the set of arcs that are both in $\mathfrak{C}_i^{\mathrm{of}}$ and its neighboring cycles. We let $\mathcal{C}^{\mathrm{of}}(e_i)$ be the set of fundamental cycles that arc $e_i \in \mathcal{E}^{\mathrm{o}}$ belongs to, i.e., $\mathcal{C}^{\mathrm{of}}(e_i) = \{\mathfrak{C}_j^{\mathrm{of}}, j \in \{1,\dots,\mu^{\mathrm{f}}\} \,|\, e_i \in \mathcal{C}_j^{\mathrm{ofe}}\}$.

2.2.3 Eliminating affine equality constraints in optimization problems

Consider the following optimization problem

$$\begin{aligned} \mathbf{x}^\star = \ & \underset{\mathbf{x}\in\mathbb{R}^n}{\operatorname{argmin}}\ \phi(\mathbf{x}), \quad \text{s.t.}, \\ & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \end{aligned} \tag{2.1}$$

where $\phi : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are the cost function and the inequality constraint function, respectively. Here, $\mathbf{A} \in \mathbb{R}^{p\times n}$ with $\operatorname{rank}(\mathbf{A}) = p \leq n$. To solve this problem, one way is to eliminate the equality constraints and then solve inequality constraint optimization problem [27]. A matrix $\mathbf{F} \in \mathbb{R}^{n\times(n-p)}$ and vector $\mathbf{x}_{\mathrm{p}} \in \mathbb{R}^n$ is found which parametrize affine

feasible solutions as

$$\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} = \mathbf{b}\} = \{\mathbf{F}\mathbf{z} + \mathbf{x}_{\mathrm{p}} \mid \mathbf{z} \in \mathbb{R}^{n-p}\}. \tag{2.2}$$

Here, $\mathbf{x}_{\mathrm{p}}$ is a particular solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$, and $\mathbf{F} \in \mathbb{R}^{n\times(n-p)}$ is any matrix whose range is the nullspace of $\mathbf{A}$. Then, $\mathbf{x}^\star$ in (2.1) satisfies $\mathbf{x}^\star = \mathbf{F}\mathbf{z}^\star + \mathbf{x}_{\mathrm{p}}$ where

$$\begin{aligned} \mathbf{z}^\star = &\underset{\mathbf{z}\in\mathbb{R}^{n-p}}{\operatorname{argmin}}\ \bar{\phi}(\mathbf{z}) = \phi(\mathbf{F}\mathbf{z} + \mathbf{x}_{\mathrm{p}}), \quad \text{s.t.} \\ &\mathbf{g}(\mathbf{F}\mathbf{z} + \mathbf{x}_{\mathrm{p}}) \leq \mathbf{0}. \end{aligned} \tag{2.3}$$

Notice that in comparison to (2.1), in (2.3) we not only eliminated the equality constraints but also decreased the number of optimization search variables from n to $n - p$.

2.3 Problem definition

In this section, we describe our optimal network flow problem of interest which is defined as a constrained convex optimization problem over a cyber-physical network. We start by defining the physical layer. Consider a network of n nodes where each node is connected to a subset of other nodes through some form of routes. For example, in a power network the route is a transmission line, while in a transportation network the route is the road connecting two conjunction nodes on the road map. The physical layer topology is described by a connected graph $\mathcal{G}_{\text{physic}} = (\mathcal{V}_{\text{physic}}, \mathcal{E}_{\text{physic}})$, where $|\mathcal{V}_{\text{physic}}| = n$ and $|\mathcal{E}_{\text{physic}}| = m$. The flow can travel in both directions in every route, however, we assume a pre-specified positive orientation for each route and based on it we describe the flow network in the physical layer by the oriented version of $\mathcal{G}_{\text{physic}}$, i.e, $\mathcal{G}^{\text{o}}_{\text{physic}} = (\mathcal{V}_{\text{physic}}, \mathcal{E}^{\text{o}}_{\text{physic}})$, where $\mathcal{V}_{\text{physic}} = \{v_1, \cdots, v_n\}$ is the node set and for the arc set if $(v_i, v_j) \in \mathcal{E}^{\text{o}}_{\text{physic}}$, then $(v_j, v_i) \notin \mathcal{E}^{\text{o}}_{\text{physic}}$. Every arc (route) (v_i, v_j) on the network has a capacity that is lower bounded and upper bounded by

pre-specified scalar values. This physical network transfers a flow $f_1 \in \mathbb{R}_{>0}$ from source node v_1 to sink node v_n (see the physical layer in Fig. 2.2), while respecting the routes capacities as well as conservation of the flow constraints at each node, i.e., the total inflow into each node must be equal to the total outflow from that node. There is a convex cost associated with flow across each arc. Let $\mathcal{E}^{\mathrm{o}}_{\mathrm{physic}} = \{e_1, e_2, \cdots, e_m\}$, and the flow across arc e_i be x_i. Our optimal power flow problem of interest is to find the network minimizer $\mathbf{x}^\star \in \mathbb{R}^m$ in the following optimization problem

$$\mathbf{x}^\star = \underset{\mathbf{x}\in\mathbb{R}^m}{\operatorname{argmin}}\ \phi(\mathbf{x}) = \sum_{i=1}^{m} \phi_i(x_i), \quad \text{s.t.,} \tag{2.4a}$$

$$\mathfrak{I}^{\mathrm{o}}\,\mathbf{x} = \mathbf{f}, \tag{2.4b}$$

$$\underline{\mathsf{b}}_i \leq x_i \leq \bar{\mathsf{b}}_i, \quad i \in \{1, \cdots, m\}, \tag{2.4c}$$

where $\mathbf{f} = [\mathsf{f}_1, \cdots, \mathsf{f}_n]^\top$, with $f_n = -f_1 \in \mathbb{R}$ given and $f_j = 0$ for $j = \{2, \cdots, n-1\}$. Also, for $i \in \{1, \ldots, m\}$, $\bar{\mathsf{b}}_i, \underline{\mathsf{b}}_i \in \mathbb{R}$ are given scalars, and $\phi_i : \mathbb{R} \to \mathbb{R}$ is convex function. Here, (2.4b) captures the flow conservation at nodes across the network and (2.4c) describes the arc capacity constraints.

For a given input flow f_1, the feasible set of problem (2.4) is give by

$$\mathcal{X}_{\mathrm{fe}} = \big\{\mathbf{x} \in \mathbb{R}^m \,|\, \mathfrak{I}^{\mathrm{o}}\,\mathbf{x} = \mathbf{f}, \underline{\mathsf{b}}_i \leq x_i \leq \bar{\mathsf{b}}_i,\ i \in \{1, \cdots, m\}\big\}. \tag{2.5}$$

For a given network and capacity bounds, the maximum (resp. minimum) network flow problem gives an upper and (resp. lower) bound on the admissible ranges of input flow f_1 such that the feasible set (2.5) is always non-empty. Maximum (also minimum) flow of a network can be find using the Edmonds-Krap algorithm in $\mathcal{O}(nm^2)$ in a central way [44].

As the size and, consequently, the number of the arcs in the physical network grow, the number of search variables in the optimization problem (2.4) grows with it linearly. Dis-

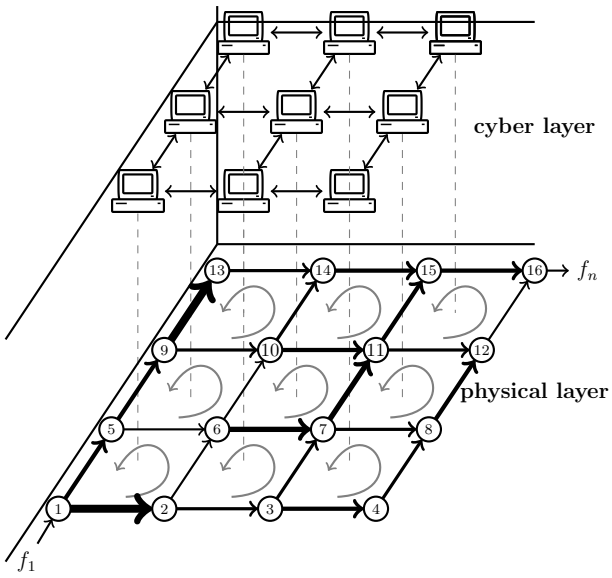


Figure 2.2: Physical and cyber layers of a optimal network flow problem with $n = 16$ and $m = 24$ arcs with different capacities in the physical layer (with different thickness). In the physical layer graph, the arrows indicate the positive flow directions. The cyber layer is constructed using the cycle basis of the physical layer graph. In the cyber layer there are $N = 9$ agents with processing and communication capabilities.

tributed solutions where agents with computation and communication capabilities solve this problem through parallel processing are proposed to solve large scale optimal network flow problems. Although, in this algorithms the computational cost of the optimization problem (2.4) does efficiently get distributed among the agents in cyber layer, the high number of decision variables normally translates to high number of agents or large communication overhead between neighboring agents.

Our first objective in this chapter is to use concepts from cycle basis and cycle flow in graph theory to reduce and obtain the minimum number of search variables for the optimal network flow problem (2.4). Our next objective is to use these results to devise a cyber layer layout with efficient communication topology to solve the optimal network flow problem (2.4) in a distributed manner using an ADMM approach.

2.4 A cycle-basis distributed ADMM solution

In this section, we address the two objectives we mentioned in Section 2.3.

2.4.1 Minimum search variable

To obtain the minimum search variable, we invoke the result discussed in Section 2.2.3 in eliminating the affine equality constraint (2.4b) from our optimal network flow problem (2.4). Our first result below, uses Theorem 2.2.1 and Theorem 2.2.2 to obtain minimum search variable for the optimal network flow problem (2.4). Even though this result gives a guideline to reduce the size of the search variable, the challenge is to obtain a matrix whose columns span the nullspace of the incidence matrix $\mathfrak{J}^{\mathrm{o}}$ in such a way that the equivalent representation of the optimization problem (2.4) is amenable to a distributed solution.

Theorem 2.4.1 (Eliminating the flow conservation constraint from (2.4))**.** *Consider the optimal network flow problem* (2.4) *over a physical network described by* $\mathcal{G}_{\mathrm{physic}} = (\mathcal{V}_{\mathrm{physic}}, \mathcal{E}_{\mathrm{physic}})$ *where* $|\mathcal{V}_{\mathrm{physic}}| = n$ *and* $|\mathcal{E}_{\mathrm{physic}}| = m$ *and its oriented graph is* $\mathcal{G}^{o}_{\mathrm{physic}}$ *. Assume that* $\mathcal{G}_{\mathrm{physic}}$ *is biconnected. Then,* $\mathbf{x}^{\star}$ *in* (2.1) *satisfies* $\mathbf{x}^{\star} = \mathbf{B}^{\mathrm{of}\,\top}\mathbf{z}^{\star} + \mathbf{x}^{\mathrm{p}}$ *where*

$$\begin{aligned} \mathbf{z}^{\star} = & \underset{\mathbf{z}\in\mathbb{R}^{\mu^f}}{\operatorname{argmin}} \phi(\mathbf{z}) = \phi(\mathbf{B}^{\mathrm{of}\,\top}\mathbf{z} + \mathbf{x}^{\mathrm{p}}), \quad s.t. \\ & b \leq \mathbf{B}^{\mathrm{of}\,\top}\mathbf{z} + \mathbf{x}^{\mathrm{p}} \leq \bar{\mathbf{b}}, \end{aligned} \tag{2.6}$$

where $\mathbf{B}^{\mathrm{of}}$ *is the oriented fundamental cycle matrix of* $\mathcal{G}^{o}_{\mathrm{physic}}$ *and* $\mathbf{x}^{\mathrm{p}}$ *is a particular solution* (2.4b).

Proof. Given the discussion in Section 2.2.3, the proof relies on showing that the nullspace of $\mathfrak{J}^{\mathrm{o}}$ is spanned by columns of $\mathbf{B}^{\mathrm{of}\,\top}$. Recall that for a connected digraph $\operatorname{rank}(\mathfrak{J}^{\mathrm{o}}) = n-1$. Therefore, the size of the nullspace of $\mathfrak{J}^{\mathrm{o}} \in \mathbb{R}^{n\times m}$ is $m-n+1$. Invoking the results of

Theorem 2.2.1 and Theorem 2.2.2, we have, respectively, $\mathfrak{J}^{\mathrm{o}}\,\mathbf{B}^{\mathrm{of}\,\top} = \mathbf{0}$ and $\mathrm{rank}(\mathbf{B}^{\mathrm{of}\,\top}) = m - n + 1$. Therefore, nullspace of $\mathfrak{J}^{\mathrm{o}}$ is spanned by columns of $\mathbf{B}^{\mathrm{of}\,\top}$. This completes our proof. □

For any given f_1, the particular solution of (2.4b) can be obtained in an efficient manner using the following result.

Lemma 2.4.1 (Particular solution of (2.4b))**.** *Let $\bar{\mathbf{x}}^p$ be a particular solution of* (2.4b) *for $\mathsf{f}_1 = 1$. Then, for any $f_1 \in \mathbb{R}$, a particular solution for* (2.4b) *is given by $\mathbf{x}^p = \mathsf{f}_1\bar{\mathbf{x}}^p$.*

Proof. Recall that in (2.4b), $\mathbf{f} = \begin{bmatrix} \mathsf{f}_1 & 0 & \cdots & 0 & \mathsf{f}_n \end{bmatrix}^\top$, where $\mathsf{f}_n = -\mathsf{f}_1$. For any $f_1 \in \mathbb{R}$, let $\mathbf{x} = \mathsf{f}_1\bar{\mathbf{x}}^{\mathrm{p}}$. Then, we can write $\mathfrak{J}^{\mathrm{o}}\,\mathbf{x} = \mathfrak{J}^{\mathrm{o}}\,\bar{\mathbf{x}}_{\mathrm{p}}\,\mathsf{f}_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 & -1 \end{bmatrix}^\top \mathsf{f}_1 = \mathbf{f}$. Therefore, a particular solution for $\mathfrak{J}^{\mathrm{o}}\,\mathbf{x} = \mathbf{f}$ is $\mathbf{x}^{\mathrm{p}} = \mathsf{f}_1\,\bar{\mathbf{x}}^{\mathrm{p}}$. □

Remark 2.4.1 (A procedure to construct particular solutions of (2.4b))**.** A particular solution for the flow conservation equation corresponding to $f_1 = 1$, $\bar{\mathbf{x}}^{\mathrm{p}}$, can be constructed using any oriented path from node v_1 to node v_n. For a given path from source node v_1 to sink node v_n, $\bar{\mathbf{x}}^{\mathrm{p}}_i = 0$ if e_i is not in this path, $\bar{\mathbf{x}}^{\mathrm{p}}_i = 1$ (resp. $\bar{\mathbf{x}}^{\mathrm{p}}_i = -1$) if the direction of e_i is along (resp. opposing) the direction from node 1 to node n (see Section 2.5 for a illustrative numerical example). To construct a sparse particular solution, one can use shortest path from node v_1 to node v_n.

2.4.2 Constructing cyber layer to solve the optimal power flow in a distributed manner

Here, we use the results of Section 2.4.1 to propose appropriate topologies for a cyber layer that can solve the optimal network flow problem (2.4), in a distributed manner, using its

equivalent minimum search variable representation (2.6). For the proceeding text, please recall our cycle basis related notations defined at the closing paragraph of Section 2.2.2.

We start by describing how every arc flow x_i, $i \in \{1, \ldots, m\}$ is expressed in terms of $\mathbf{z} \in \mathbb{R}^{\mu^{\mathrm{f}}}$. Given that $\mathbf{x} = \mathbf{B}^{\mathrm{of}\,\top}\mathbf{z} + \mathbf{x}^{\mathrm{p}}$, then $x_i = \mathbf{z}^{\top}[\mathbf{B}^{\mathrm{of}}]_i + \mathsf{x}_i^{\mathrm{p}}$, where $[\mathbf{B}^{\mathrm{of}}]_i$, $i \in \{1, \ldots, m\}$, is the i^{th} column of $\mathbf{B}^{\mathrm{of}}$. Notice that one can think of every z_i, $i \in \{1, \ldots, \mu^{\mathrm{f}}\}$ as a cycle flow variable (with positive direction in clockwise direction) of the fundamental cycle $\mathfrak{C}_i^{\mathrm{of}}$. Recall that, for a given arc $e_i \in \mathcal{E}^{\mathrm{o}}_{\mathrm{physic}}$, every element of $\mathbf{B}^{\mathrm{of}}_{ji}$ is zero except if cycle $\mathfrak{C}_j^{\mathrm{of}}$ contains arc e_i, i.e., $e_i \in \mathcal{C}_j^{\mathrm{ofe}}$. As a result, we can deduce that every x_i, $i \in \{1, \ldots, m\}$, is an affine function of $\mathsf{x}_i^{\mathrm{p}}$ and $\{z_k\}_{k \in \{j \in \{1,\ldots,\mu^{\mathrm{f}}\} \,|\, e_i \in \mathcal{C}_j^{\mathrm{ofe}}\}}$, indicating that every arc flow is a function of its corresponding element of the particular solution and the cycle flow of fundamental cycles that contain the arc. Given such relationship, then the cost function of every arc is

$$\begin{aligned}\phi_i(x_i) &= \phi_i(\mathbf{z}^{\top}[\mathbf{B}^{\mathrm{of}}]_i + \mathsf{x}_i^{\mathrm{p}}) \\ &= \psi_i(\{z_k\}_{k \in \left\{j \in \{1,\ldots,\mu^{\mathrm{f}}\} \,|\, e_i \in \mathcal{C}_j^{\mathrm{ofe}}\right\}}, \mathsf{x}_i^{\mathrm{p}}).\end{aligned} \tag{2.7}$$

Based on the observation above, we propose a structure for the cyber layer to solve the optimal network flow problem which we describe below.

Cyber layer architecture: we propose to assign a cyber layer node to each fundamental cycle (see Fig. 2.2 as an example). We assume that the cyber layer nodes of neighboring fundamental cycles can communicate with each other in bidirectional way. For biconnected physical layer graphs this procedure will result in a connected graph of $\mu^{\mathrm{f}} = m - n + 1$ nodes for cyber layer (see Fig. 2.2 and Fig. 2.3 for examples).

To simplify our notation for distributed ADMM solver, we assume that the physical layer graph is planar and among the set of cycle basis we choose the one that each arc belongs to at most two fundamental cycles (c.f. Mac Lane's planarity criterion [85]).

Next, we describe how the optimal cost flow problem (2.4) can be cast in a separable manner among the cyber layer nodes in a way that the problem can be solved by implementing a distributed partial variable ADMM algorithm [26, 88]. To this end, for every cyber layer node $i \in \{1, \ldots, \mu^{\mathrm{f}}\}$, we define $\mathbf{y}_i = (\bar{y}_i, \tilde{\mathbf{y}}_i) \in \mathbb{R}^{|\mathcal{N}_i^{\mathfrak{C}}|+1}$, where $\bar{y}_i$ is the local copy of z_i (the cycle flow of the fundamental cycle corresponding to cyber node i) and $\tilde{\mathbf{y}}_i$ is the local copy of $\{z_k\}_{k \in \mathcal{N}_i^{\mathfrak{C}}}$ (the cycle flows of the neighboring fundamental cycles of cyber node i) at cyber node i. Next, we cast the cost function of each cyber node in terms of its decision variable $\mathbf{y}_i$. For every cyber layer node, we define its cost function as the sum of costs of its exclusive member arcs, i.e., the arcs that are in $\bar{\mathcal{C}}^{\mathrm{ofe}}$ (cost function of these nodes depend on only cycle flow z_i of cyber node i) plus sum of 0.5 costs of arcs that it shares with its neighbors, i.e., the arcs that are in $\tilde{\mathcal{C}}^{\mathrm{ofe}}$ (cost function of these nodes depend on the cycle flow of the cyber node i and the flow of the neighbor node containing them). Recall that we assumed that every cyber node has a copy of the cycle flow variables of its neighbors. We represent the cost of each cyber layer node i by $\theta_i(\mathbf{y}_i)$ as follows (recall (2.7))

$$\theta_i(\mathbf{y}_i) = \bar{\theta}_i(\bar{y}_i) + \tilde{\theta}(\bar{y}_i, \tilde{\mathbf{y}}_i), \tag{2.8}$$

where

$$\bar{\theta}_i(\bar{y}_i) = \sum_{\forall e_k \in \bar{\mathcal{C}}_i^{\mathrm{ofe}}} \psi_k(\bar{y}_i, \mathsf{x}_k^{\mathrm{p}}),$$

and

$$\tilde{\theta}(\bar{y}_i, \tilde{\mathbf{y}}_i) = \frac{1}{2} \sum_{\forall e_k \in \tilde{\mathcal{C}}_i^{\mathrm{ofe}}} \psi_k(\bar{y}_i, [\tilde{\mathbf{y}}_i]_{e_k}, \mathsf{x}_k^{\mathrm{p}}),$$

where $[\tilde{\mathbf{y}}_i]_{e_k}$ gives the component of $\tilde{\mathbf{y}}_i$ that corresponds to the local copy of the cycle flow of

the neighboring cycle which contains e_k. Now we cast the optimal network flow problem in the following equivalent form that can be solved by the cyber layer nodes using a distributed ADMM method

$$
\begin{aligned}
&\mathbf{y}^\star = \underset{\mathbf{y}_1,\cdots,\mathbf{y}_{\mu^{\mathrm{f}}}}{\operatorname{argmin}} \sum_{i}^{\mu^{\mathrm{f}}} \theta_i(\mathbf{y}_i) \qquad \text{s.t.} \\
&\text{for} \;\; i \in \{1,\ldots,\mu^{\mathrm{f}}\} : \\
&\bar{y}_i = \hat{y}_i, \\
&[\tilde{\mathbf{y}}_i]_{e_k} = \hat{y}_j, \qquad\qquad\qquad \forall e_k \in \tilde{\mathcal{C}}_i^{\mathrm{ocf}}, j = [\mathcal{N}_i^{\mathfrak{C}}]_{e_k}, \\
&\underline{\mathsf{b}}_k \leq \mathbf{B}_{ik}^{\mathrm{of}} \bar{y}_i + \mathbf{B}_{jk}^{\mathrm{of}} [\tilde{\mathbf{y}}_i]_{e_k} + \mathsf{x}_k^{\mathrm{p}} \leq \bar{\mathsf{b}}_k, \quad \forall e_k \in \tilde{\mathcal{C}}_i^{\mathrm{ocf}}, j = [\mathcal{N}_i^{\mathfrak{C}}]_{e_k}, \\
&\underline{\mathsf{b}}_k \leq \mathbf{B}_{ik}^{\mathrm{of}} \, \bar{y}_i + \mathsf{x}_k^{\mathrm{p}} \leq \bar{\mathsf{b}}_k, \qquad\qquad \forall e_k \in \bar{\mathcal{C}}_i^{\mathrm{ocf}},
\end{aligned}
\tag{2.9}
$$

where $[\mathcal{N}_i^{\mathfrak{C}}]_{e_k}$ gives the cycle neighbor of node i which also contains arc e_k. Recall that because we assume the graph is planar, for every cyber/cycle node i, for each given $e_k \in \tilde{\mathcal{C}}_i^{\mathrm{ocf}}$, there is only one neighbor that contains that arc. Here, the primal variable of each agent in the ADMM method at each agent $i \in \{1,\ldots,\mu^{\mathrm{f}}\}$ is $\mathbf{y}_i = (\bar{y}_i, \tilde{\mathbf{y}}_i)$. And, the auxiliary variables of the ADMM method are $\{\hat{y}_i\}_{i\in\{1,\ldots,\mu^{\mathrm{f}}\}}$. These augxillary variables, through the affine conditions in (2.9), enforce the local copy of any cycle flow at a cyber node to be the same as the local copies of its neighbors.

Here, we assume that every cyber node knows the components of the particular solution associated with the arcs constituting its corresponding fundamental cycle basis. Once every cyber node $i \in \{1,\ldots,\mu^{\mathrm{f}}\}$ computes $\mathbf{y}_i^\star$, its component of the optimal solution $\mathbf{y}^\star$, then it can proceed with

$$
\begin{aligned}
&x_k^\star = \mathbf{B}_{ik}^{\mathrm{of}} \, \bar{y}_i + \mathsf{x}_k^{\mathrm{p}}, \qquad\qquad \forall e_k \in \bar{\mathcal{C}}_i^{\mathrm{ocf}} \\
&x_k^\star = \mathbf{B}_{ik}^{\mathrm{of}} \, \bar{y}_i + \mathbf{B}_{jk}^{\mathrm{of}} \, [\tilde{\mathbf{y}}_i]_{e_k} + \mathsf{x}_k^{\mathrm{p}}, \quad \forall e_k \in \tilde{\mathcal{C}}_i^{\mathrm{ocf}}, j = [\mathcal{N}_i^{\mathfrak{C}}]_{e_k},
\end{aligned}
$$

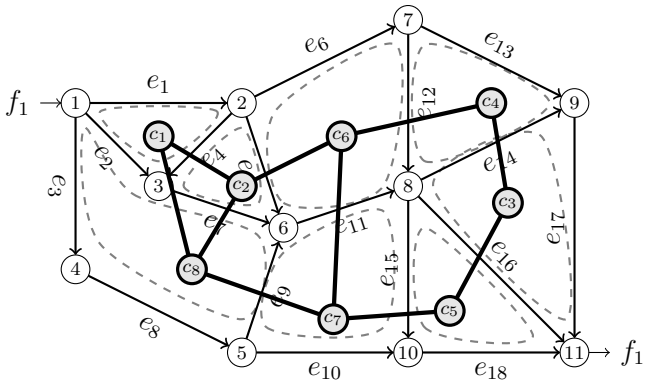


Figure 2.3: Oriented version of biconnected graph of Fig 2.1 which represents the physical layer in our numerical example. The inflow is through node v_1 which leaves the network through node v_{11}. The highlighted graph in bold is the cyber layer graph constructed from the cycle basis shown in Fig 2.1.

to obtain the optimal cycle flows through its arcs, i.e., the optimal flow through $\forall e_k \in \mathcal{C}_i^{\text{ocf}}$.

For networks with large fundamental cycle sizes, one can split a cycle among several cyber nodes. In this case the length of the $\bar{y}_i$ of these agents will be 0 and we can still use the distributed ADMM algorithm (c.f. [26] and[88]) to solve the problem. Similarly, two or more cycles can be assigned to one cyber layer. The details are omitted for brevity.

Also notice that once the cyber layer and the particular solution $\bar{\mathbf{x}}^{\text{p}}$ are established, they stay the same for all input flow f_1.

2.5 Numerical example

In this section, we use a numerical example to demonstrate the effectiveness of our cycle basis distributed solution in solving an optimal network flow problem. We use the network in Fig. 2.1 as our physical layer network. We assign positive flow orientation to the arcs as represented in Fig. 2.3. We assume the lower and upper bound capacities satisfy $\underline{\mathsf{b}}_i = -\bar{\mathsf{b}}_i$, $i \in \{1, \ldots, 18\}$, where we pick $\bar{\mathsf{b}}_i$ uniformly randomly from $[2, 50]$. Here, the cost of the network flow x_i through each arc e_i, $i \in \{1, \ldots, 18\}$, is given by $\phi_i(x_i) = (x_i/\bar{\mathsf{b}}_i)^2$.

We use Matlab 'quadprog' to solve the problem in a centralized manner to generate reference values to compare the performance of our distributed cycle basis distributed ADMM algorithm as outlined in Section 2.4. For the given capacities, the maximum flow is 59, i.e., the external flow should be in the admissible range of $[-59, +59]$.

Recall that using the cycle basis method, we have $x_i = \mathbf{z}^\top [\mathbf{B}^{\text{of}}]_i + \mathsf{x}_i^{\text{p}}$, $i \in \{1, \ldots, 18\}$, where for network of Fig. 2.1 the cycle flow vector is $\mathbf{z}^\top = [z_1, ..., z_8]$. Therefore, using the cycle basis method, the number of the decision variables of the optimal flow problem (2.4), here, reduces from 18 to 8 in (2.6). For the network of Fig. 2.1, the fundamental orientated cycle basis matrix is (recall that we have assumed the positive cycle flow direction to be clockwise)

$$\mathbf{B}^{\text{of}} = \begin{bmatrix}
1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

To obtain the particular solution for our cycle basis optimal network flow solver, we invoke the results discussed in Lemma 2.4.1 and Remark 2.4.1. First, we construct the particular solution corresponding to $f_1 = 1$, $\bar{\mathbf{x}}^{\text{p}}$, using the shortest path $\{e_1, e_5, e_{11}, e_{16}\}$ from source node v_1 to sink node v_{11} in the physical layer graph. Here, this solution is $\bar{\mathbf{x}}^{\text{p}} = (1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0)$ (recall that the choice of shortest path is to

obtain a sparse solution). This particular solution is for $f_1 = 1$, and for any other external flow f, the new particular solution is $\mathbf{x} = \mathsf{f}\,\bar{\mathbf{x}}^{\mathrm{p}}$. In other words, for any external flow, the particular solution vector $\bar{\mathbf{x}}^{\mathrm{p}}$ is for a same shortest path, and we only multiply $\bar{\mathbf{x}}^{\mathrm{p}}$ by the external flow value.

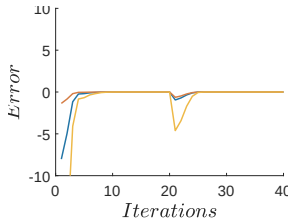
To solve our network flow problem in a distributed manner, we generate the cyber layer based on the minimum weight cycle basis shown in Fig. 2.1. This cyber layer solves the problem using the distributed ADMM solution described in (2.9). For our simulation we consider a scenario where the input flow $f_1 = 59$ for 50 iteration and it is changes to $f_1 = 30$ afterwards. Fig. 2.4 depicts our simulation result.

In Fig. 2.4, plots show the distance of arc flow values from their optimum solution during execution of distributed ADMM. During the first 50 iteration the distributed ADMM converges to the optimum solution. Then, for the second external flow, it converges to the optimum solution again. The results illustrate that for different values of external flow, the distributed ADMM converges to optimum solution using the same cycle basis matrix and particular solutions that are generated by scaling the same $\bar{\mathbf{x}}^{\mathrm{p}}$.

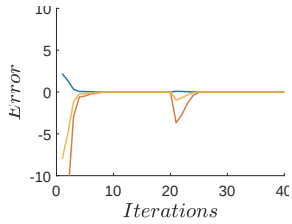
2.6 Conclusion and future research

We consider the optimal network flow problem and investigated how the decision variables of this problem can be reduced from m variables for a physical network of n nodes and m arcs to $m - n + 1$ variables for biconnected graphs. The results of this project are presented in [12] and [11]. Our study was based on exploiting the cycle basis concept from graph theory. We also proposed a new cyber layer architecture which solves our equivalent reduced search variable representation of the optimal network flow problem in a distributed manned using an ADMM approach. Our future work includes a study of dynamic optimal flow problem where

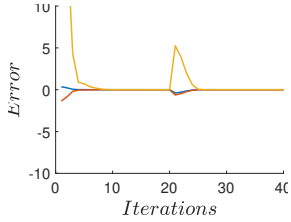
the throughput flow changes with time. We will also explore extensions of our algorithm to solve problems with multiple sources and multiple sinks.



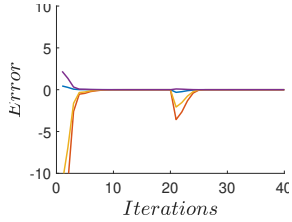
(a) cyber layer node 1 (fundamental cycle 1 consisted of arcs $\{e_1, e_2, e_4\}$)



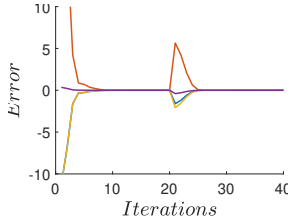
(b) cyber layer node 2 (fundamental cycle 2 consisted of arcs $\{e_4, e_5, e_7\}$)



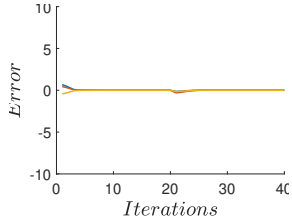
(c) cyber layer node 3 (fundamental cycle 3 consisted of arcs $\{e_{14}, e_{16}, e_{17}\}$)



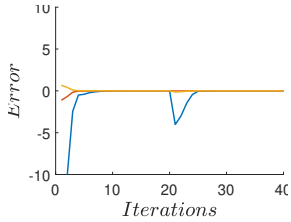
(d) cyber layer node 4 (fundamental cycle 4 consisted of arcs $\{e_{12}, e_{13}, e_{14}\}$)



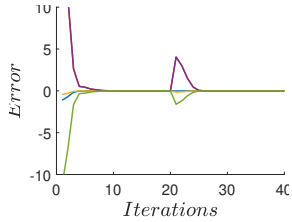
(e) cyber layer node 5 (fundamental cycle 5 consisted of arcs $\{e_{15}, e_{16}, e_{18}\}$)



(f) cyber layer node 6 (fundamental cycle 6 consisted of arcs $\{e_5, e_6, e_{11}, e_{12}\}$)



(g) cyber layer node 7 (fundamental cycle 7 consisted of arcs $\{e_9, e_{10}, e_{11}, e_{15}\}$)



(h) cyber layer node 8 (fundamental cycle 8 consisted of arcs $\{e_2, e_3, e_7, e_8, e_9\}$)

Figure 2.4: Each plot depicts $x_i(k) - x_i^\star$, for e_i's in that sub-captioned cyber layer node (fundamental cycle). Cyber layer nodes use the distributed ADMM algorithm outlined in Section 2.4 to solve the problem. We use Matlab 'quadprog' to solve the problem in a centralized manner to generate reference values $x_i^\star$, $i \in \{1, \ldots, 18\}$. As the plots show, every cyber node asymptotically calculates the optimal arc flow for its arcs.

Chapter 3

Spatio-temporal clustering of traffic data

Traffic data is a challenging spatio-temporal data, and a multi-variate time series data with spatial similarities. Clustering of traffic data is a fundamental tool for various machine learning tasks including anomaly detection, missing data imputation and short term forecasting problems. In this chapter, first, we formulate a spatio-temporal clustering problem and define temporal and spatial clusters. Then, we propose an approach for finding temporal and spatial clusters with a deep embedded clustering model. The proposed approach is examined on traffic flow data. In the analysis, we present the properties of clusters and patterns in the dataset. The analysis shows that the temporal and spatial clusters have meaningful relationships with temporal and spatial patterns in traffic data, and the clustering method effectively finds similarities in traffic data. The results of the work is published in [15].

3.1 Introduction

The volume of traffic flow data has increased with the advent of new sensing technologies. In most transportation systems, a large number of sensors gather traffic data in a geographical area. The data are gathered over specific time periods, and it has both spatial and temporal patterns. Traffic data is considered a high-dimensional time series data, which include the flow, speed and occupancy of a large number of sensors, and in which there is a spatial correlation among neighboring time series. The problem is challenging because of both temporal and spatial patterns in data. Spatio-temporal data arise in broad areas of engineering and environmental sciences. In [17], they discuss various types of spatio-temporal data and their corresponding data mining approaches. In transportation systems, traffic flow data is a challenging spatio-temporal data. Analyzing such data can improve the performance of intelligent transportation systems, reduce traffic congestion, and air pollution [91]. Also, spatio-temporal congestion patterns is studied for traffic flow data [106]. However, the increasing volume of traffic data requires the development of large-scale machine learning algorithms and big data analytics [151]. Various studies examine data-driven approaches and large-scale machine learning techniques in transportation systems [128].

Exploring similarities in spatio-temporal patterns results in finding interesting patterns in data. Clustering algorithms not only find similarities among data points, but also explore deep insights in temporal and spatial patterns. Clustering algorithms have been used in a broad range of machine learning tasks. In this chapter, we focus on developing a clustering method for spatio-temporal data. Here, we describe some of the important applications of using clustering method to solve machine learning problems. In [5], a clustering method is used for detecting anomalies. Data points which are far from center of clusters or are in separate clusters can be considered as anomalies. In [67], they use k-means clustering for finding similar spatial data. Then an autoencoder imputes missing data for each cluster. In [16] and [29], the spatial clusters are used to improve the performance of a forecasting

problem. These works illustrate that the clustering of spatio-temporal data is a fundamental tool for various machine learning tasks.

Moreover, clustering algorithms have been used on various types of traffic flow data. In [34], they cluster vehicle trajectory data to model traffic flow. The data is obtained from traffic surveillance of intersections. In [31], they cluster traffic flow patterns based on dynamical traffic flow models. A macroscopic flow model is used with a multivariate clustering approach. However, most of traffic flow data are obtained by loop detectors or sampling from GPS data on road networks. Each road segment includes traffic data for a time stamp. Here, we focus on traffic data obtained by loop detectors. Clustering of traffic data is a common data mining analysis [61]. In [36], they propose a fuzzy clustering technique for traffic flow data. A Dynamic Time Warping (DTW) distance among traffic flow data is used as a similarity measure. Also, clustering of traffic data can find interesting insights in physical properties of traffic flow data. In [115], they apply fuzzy k-means clustering to analyze variations in traffic flow states. While clustering of traffic data is an important problem, it is a challenging one. Traffic data is spatio-temporal data in which there is a non-linear relation among spatial and temporal neighborhoods. Clustering of time series data requires a distance metric, such as DTW, which extracts a non-linear relation among time series data. Moreover, traffic data is a multi-variate time series with spatial correlation. Applying traditional clustering methods, such as k-means, on traffic data is computationally expensive and can have low performance [54], though more efficient heuristic methods for k-means clustering of traffic flow data have been studied [118].

Since traffic flow data is a multi-variate time series data, here we review some of the recent works for time series clustering. In [3], they describe a broad range of time series clustering applications. Also, the main components of time series clustering have been studied, including time series representations, similarity and distance measures, clustering prototypes and time series clustering. In [117], they describe the challenges of k-means clustering with

time warp measure. They propose a weighted and kernel time warp measures for k-means clustering. Their method has a faster estimation of clusters. Further investigation in time series clustering is studied in [96].

Deep learning models significantly improve performance of various machine learning problems, such as computer vision and natural language processing. Such models have been broadly used for various large-scale spatio-temporal problems [132], [13]. Moreover, deep learning models for clustering tasks are broadly studied in [87]. Deep embedded clustering is primarily introduced in [136]. Variations of the model have been studied in a broad domains. A joint training of the model to preserve the latent feature space structure is proposed [48]. In [138], they analyze a clustering-friendly latent representations, in which jointly optimize dimension reduction by a neural network and a k-means clustering. While most of the works apply deep embedded clustering on images, there are few studies to show their performance on time series data. In [125], they jointly cluster and train the model. They also segment time series data with agglomerative clustering.

The aforementioned works show the importance of developing a deep learning model for the spatio-temporal clustering problem. In addition, they show the recent advances in deep learning models for clustering tasks. However, there is lack of study to investigate the performance of deep learning models for spatio-temporal clustering. Here, we formulate and propose a procedure for spatio-temporal clustering with deep learning models. We define temporal and spatial clusters and analyze spatial and temporal clusters obtained by deep learning models in details. Such an analysis shows the existence of temporal and spatial patterns in the clusters. Also, we illustrate that the proposed approach can better finds temporal clusters, as it captures the non-linear relation among temporal data. Our focus in this study is on traffic flow data. Clustering of such data not only improves the performance of clustering based machine learning models, such as anomaly detection with a clustering method, but also provides opportunities to further investigate transportation systems using

temporal and spatial clusters.

In section 2, we describe the technical background for the proposed approach. In section 3, we define spatial and temporal clusters, and describe the proposed approach for finding spatial clusters with temporal clusters. In section 4, a deep learning investigation of clustering on traffic flow data is illustrated. In Section 5 our conclusion and future work are presented.

3.2 Technical background

3.2.1 Problem definition

Spatio-temporal data is represented with a matrix $\mathbf{X} \in \mathbb{R}^{s \times \bar{t} \times f}$, where s is the number of sensors, $\bar{t}$ is the number of time stamps and f is the number of traffic features, including flow, speed and occupancy. Each sensor has its own time series data $\mathbf{x}_i \in \mathbb{R}^{\bar{t} \times f}$. Clustering of whole time series is computationally expensive and finding temporal patterns in such clusters is not possible. Hence, we generate sub-sequences of time series data and apply clustering methods on them. A sliding window approach with time window w generates a sequence of traffic data. A traffic state at time stamp t and location i is represented with $\mathbf{X}_i^t \in \mathbb{R}^{w \times f}$. In other words, the data points are represented with traffic states.

A clustering method assigns each traffic state $\mathbf{x}_i^t$ into a cluster $\mathbf{c}_j$, where $j \in \{1, \ldots, |C|\}$, and $|C|$ is the given number of clusters. For a latent representation of $\mathbf{x}_i^t$, represented with $\mathbf{z}_i^t$, the clustering method assigns $\mathbf{z}_i^t$ to the clusters. A temporal cluster consists of traffic states from all sensors. In other words, any traffic state can be a member of a temporal cluster. In algorithm 1, we find spatial clusters from temporal clusters. A spatial cluster includes some of the sensors. A spatial cluster is obtained based on similarity of temporal clusters. The similarity of two sensors is obtained based on the similarity of their temporal clusters. In

this way, instead of comparing two sensors' data for a long time period, e.g. six of months of training data, we have temporal similarity for a short time windows with size w.

Similarity of time series cannot be obtained from the euclidean distance of their vectors. A non-linear distance function, e.g. DTW, can provide a similarity (distance) measure for two time series. Another approach is to find latent feature representations and use a euclidean distance measure on those representations, as the temporal relation does not necessarily hold in lower dimension space. We consider both approaches in this chapter. We also examine this property of latent features in experimental results by finding the correlation between DTW distance of traffic states and euclidean distance of latent features. In such a way, the clustering is more efficient, as the distance function is applied on lower dimension space.

3.2.2 Autoencoders

An autoencoder is primarily proposed in [127]. It reconstructs the input data with $\bar{\mathbf{x}} = \mathsf{f}(\mathbf{x}, \boldsymbol{\theta})$, given model parameters $\boldsymbol{\theta}$, an input data $\mathbf{x}$ and reconstructed input $\bar{\mathbf{x}}$. An encoder is the first neural network component, which reduces the dimension of input data to a latent feature space $\mathbf{h} \in \mathbb{R}^{\mathsf{d}}$, where $\mathsf{d} < \mathsf{n}$. Latent features consists of the most important patterns of input data. The second neural network component is a decoder, which reconstructs the input data from its latent representation.

$$\mathbf{h} = \sigma(\mathsf{drop}(\mathbf{x})\mathbf{w}^1 + \mathbf{b}^1) \tag{3.1}$$

$$\bar{\mathbf{x}} = \sigma(\mathsf{drop}(\mathbf{h})\mathbf{w}^2 + \mathbf{b}^2) \tag{3.2}$$

where the activation function and the dropout function are represented with $\sigma(.)$ and $\mathsf{drop}(.)$, respectively. In a deep autoencdoer, the encoder considers several layers and reduces the

input dimension into latent feature space $\mathbf{h}$. Then the decoder as a multi layer neural network, reconstructs the input. The loss function $\mathsf{L}(\mathbf{x}, \bar{\mathbf{x}})$, e.g. mean square error, reduces the difference of input data $\mathbf{x}$ and its reconstruction $\bar{\mathbf{x}}$.

3.2.3 Deep embedded clustering

A Deep embedded clustering neural network is introduced in [136]. The encoder transforms $\mathbf{x}$ into latent feature space $\mathbf{z}$. The clustering layer is connected to latent feature layer. The weights of clustering layer is initialized with cluster centers obtained by k-means clustering. Cluster center i is represented with $\mu_{\mathsf{i}} \in \mathbb{R}^{\mathsf{d}}$. Given k as the number of clusters, and d as latent feature size, the clustering layer is represented with a dense layer $\mathbb{R}^{\mathsf{d}} \rightarrow \mathbb{R}^{\mathsf{k}}$. In other words, it converts latent features into a vector of size k, which k-th element represents the probability that the data point is assigned to the cluster k.

Given initial cluster centers $\{\mu_1, \ldots, \mu_k\}$ and latent features $\mathbf{z}$, a student's t-distribution measures the similarity between cluster centers μ_{i} and data points $\mathbf{x}_{\mathsf{i}}$ as follows,

$$\mathsf{q}_{\mathsf{ij}} = \frac{(1 + ||\mathsf{z}_{\mathsf{i}} - \mu_{\mathsf{j}}||^2)^{-1}}{\sum_{\mathsf{k}}(1 + ||\mathsf{z}_{\mathsf{i}} - \mu_{\mathsf{k}}||^2)^{-1}} \tag{3.3}$$

where the degree of freedom of the student's t-distribution is one. The probability of assigning data point $\mathbf{x}_{\mathsf{i}}$ to cluster μ_{j} is represented with q_{ij}. The assigned cluster is $\text{argmax}_{\mathsf{j}}\, \mathbf{q}_{\mathsf{ij}}$. The clustering algorithm iteratively adjusts clusters by learning from high confidence assignments. To learn from high confidence assignments, an auxiliary target distribution p_{ij} is as follows,

$$\mathsf{p}_{\mathsf{ij}} = \frac{\mathsf{q}_{\mathsf{ij}}^2/\mathsf{f}_{\mathsf{j}}}{\sum_{\mathsf{k}} \mathsf{q}_{\mathsf{ij}}^2/\mathsf{f}_{\mathsf{k}}}, \tag{3.4}$$

KL-divergence loss between q_{ij} and p_{ij} learns the high confidence soft cluster assignment,

$$KL(P||Q) = \sum_{i} \sum_{j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{3.5}$$

A joint training of autoencoders with reconstruction loss and clustering loss is proposed in [48], in which an autoencoder reconstructs input and cluster data points simultaneously. A joint training preserves the structure of latent feature representation of data points. In this chapter, we pretrain an autoencoder with traffic states and using a joint training method we obtain the clusters of temporal data.

3.3 Spatio-temporal clustering

Here, we describe the proposed method for spatio-temporal clustering of traffic data.

As represented in Section 2.1, we consider the clustering of traffic states $\mathbf{x}_i^t \in \mathbb{R}^{w \times f}$ for a given time window w and features f. While in alternative approaches, one can consider the problem of clustering a whole time series $\mathbf{x}_i \in \mathbb{R}^{\bar{t} \times f}$, or sub sequences of spatio-temporal data $\mathbf{x}^t \in \mathbb{R}^{s \times w \times f}$. Clustering of traffic states provides more flexibility and efficiency. First, we can have multiple clusters for one sensor in different time stamps, unlike clustering of whole time series. Second, we can cluster sensors independent of one other, unlike clustering of sub sequences of spatio-temporal data. However, in this approach, we need a method to obtain temporal and spatial clusters. In this section, we describe our proposed method.

The input of autoencoder consists of all traffic states $\mathbf{x}_i^t$ for all sensors i and time stamps t. The autoencoder reconstructs traffic states. Also, the encoder outputs their latent feature representations. The deep embedded clustering assigns the latent feature of each traffic state to the clusters. The probability that each traffic state would be in cluster k is represented

with c_{ik}^t. The assigned cluster would be obtained by $c_i^t = \text{argmax}_k\, c_{ik}^t$. We represent the temporal cluster c_i^t for sensor i and time stamp t. Although temporal clusters can find meaningful patterns in data, spatial clustering is useful to find the relation of sensors in a geographical area. Spatial clustering needs aggregation of temporal clusters. First, we define spatial similarity as the similarity of temporal clusters for a time interval. In other words, spatial similarity of two sensors i and j for a time time interval $\mathbf{T}$ is the number of times their temporal clusters are equal for $t \in \mathbf{T}$. For example, if the objective is to find spatial similarity of sensors between 3pm to 5pm on one day, then it can be obtained by comparing c_i^t and c_j^t for all t in range of 3pm to 5pm. In this approach, the spatial clustering is dynamic over various time intervals. Spatial clustering can be obtained based on these temporal clusters. In the rest of this section we describe the proposed procedure.

The proposed procedure is in Algorithm 5.1. The input of spatio-temporal data is represented with $\mathbf{x} \in \mathbb{R}^{s \times \bar{t} \times k}$. First we normalize data with $\mathsf{Normalize}(.)$. All of the features are normalized in the range of zero to one over time stamps to have a same maximum and minimum value. A sliding window method $\mathsf{SlidingWindow}(.,.)$ receives input data and window size of w. A sequence of $\mathbf{x}^t \in \mathbb{R}^{s \times w \times k}$ is generated for $\{0, \ldots, \bar{t}\}$. An autoencoder $\mathsf{trainAutoencoder}(.)$ receives an input traffic states $\mathbf{x}_i^t$ for all sensors i and time stamp t. It reconstructs each input data and train model parameters. The model parameters are stored in $\mathsf{modelPre}$. Since the autoencoder has encoder-decoder architecture, it can output the latent feature representation of input data with $\mathsf{predictLatent}(.)$, as the output of layer $\mathbf{z}$ with lowest dimension layer.

The first step of applying deep embedded clustering is to find initial clusters as introduced in [136]. A k-means clustering $\mathsf{initialKmeans}(.)$ is applied on latent feature representations for few iterations. The output is the initial cluster means. For a given number of clusters c, the mean of clusters is found, represented with $\boldsymbol{\mu}_k$ for cluster k. The deep embedded clustering $\mathsf{trainEmbeddedC}(.,.,.)$ receives the pre-trained autoencoder, input data and mean of clusters. The model has one clustering layer connected to latent feature layer, further

Table 3.1: Clustering of spatio-temporal data

1: **procedure** CLUSTERING($\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_s\}$) ▷ Spatio-temporal data with s sensors
2: $\mathbf{X} \leftarrow \mathsf{Normalize}(\mathbf{X})$ ▷ Normalize each sensor
3: $[\mathbf{x}^1, \ldots, \mathbf{x}^{\bar{t}}] = \mathsf{slidingWindow}(\mathbf{X}, w)$ ▷ Generating sequence of time windows
4: $\mathsf{modelPre} = \mathsf{trainAutoencoder}([\mathbf{x}^1, \ldots, \mathbf{x}^{\bar{t}}])$
5: $[\mathbf{z}^1 \ldots, \mathbf{z}^t] = \mathsf{modelPre.predictLatent}([\mathbf{x}^1, \ldots, \mathbf{x}^{\bar{t}}])$
6: $[\mu^1, \ldots, \mu^c] = \mathsf{initialKmeans}([\mathbf{z}^1, \ldots, \mathbf{z}^{\bar{t}}])$
7: $\mathsf{modelEmb} = \mathsf{trainEmbC}(\mathsf{modelPre}, [\mathbf{x}^1, \ldots, \mathbf{x}^{\bar{t}}], [\mu^1, \ldots, \mu^c])$
8: ▷ End of training step.
9: $\mathbf{Q} = \mathsf{modelEmb.predictClusters}([\mathbf{x}^1, \ldots, \mathbf{x}^{\bar{t}}])$
10: **for** $i \leftarrow 1$ to s
11: **for** $j \leftarrow i$ to s
12: $\mathbf{C}[i, j] = \sum_{t \in \mathbf{T}} (\mathbf{Q}^t_i == \mathbf{Q}^t_j)$
do **do**
13: **return** $\mathbf{C}$ ▷ Return the similarity of sensors in dataset.

details were provided in Section 2.3 of this chapter. The output of deep embedded clustering is $\mathbf{Q} \in \mathbb{R}^{\bar{t} \times s, c}$, which includes the probability of membership of each traffic state $\mathbf{x}^t_i$ to the clusters c clusters. For all sensors in dataset, the number of times two sensors are in same clusters is considered as their spatial similarity, which is obtained in lines 9-13 of Algorithm 1. This comparison is for all t in time interval $\mathbf{T}$. In other words, for each time interval, we can obtain spatial similarity among sensors. Such a value is stored in $\mathbf{C}$. Since the value of similarity matrix $\mathbf{C^T}$ depends on time interval $\mathbf{T}$, we can obtain different spatial similarities over various times of day. To find spatial clusters from $\mathbf{C^T}$, a threshold value ϵ on each element $\mathbf{C^T_{ij}}$ can finds the clusters. In other words, if we have $\mathbf{C^T_{ij}} > \epsilon$, then two sensors i and j are in a same cluster. In the experimental results, we illustrate the output similarity matrix $\mathbf{C}$ using heatmaps and spatial clusters obtained by Algorithm 1. This approach for finding spatial clusters has advantages over applying a clustering method on spatio-temporal data directly. Our objective is to find a dynamic spatial cluster set. In such a way, we can obtain various spatial clustering in different time intervals.

3.4 Experimental results

Here we illustrate the results for clustering of traffic flow data. The objective is to show the existence of spatial and temporal patterns in the found clusters.

The deep learning model is implemented with Keras. We use a fully-connected autoneocder with 7 layers. All of the layers have Relu activation function and dropout rate of 0.2. The number of hidden units are $(32, 32, 128, 4, 128, 32, 32)$ in seven fully-connected layers. The batch size of 288, one day with 5-min time stamp, and Adam optimizer are selected. The deep embedded clustering layer has a l_2 regularization of 0.1. The loss function in joint training of the model is 0.05 for clustering loss term, the KL divergence of clustering layer's output and auxiliary target, and 1.0 for reconstruction loss term. The higher value of reconstruction loss keeps the structure of latent feature space. The joint training of clustering layer with reconstruction loss has 5000 epochs, and every 50 epochs we update the auxiliary target distribution $\mathbf{p}$ with high confidence soft cluster assignments.

3.4.1 Dataset

Traffic data are selected from the PeMS data[1], which have widely used by researchers examining large-scale traffic flow models. Traffic data is gathered from main-line loop detector sensors every 30 seconds and aggregated to every 5 minutes. The data is for US-101 South highway, in the Bay Area of California, which includes 26 mainline sensors, illustrated in Fig. 3.1. We select the data for the first two months of 2016. The data includes flow, speed and occupancy. In a preprocessing step, we re-scale data into the range of $[0, 1]$. Since the data is not stationary, the output of clustering can be meaningless and dependent on the hours of a day. We subtract each time window of size w from its first element. After hyper parameter tuning, a time window of size 12, one hour, is selected. In such a way, we expect

the model to cluster time series data based on the slope, periodic patterns and non-linear temporal similarities.

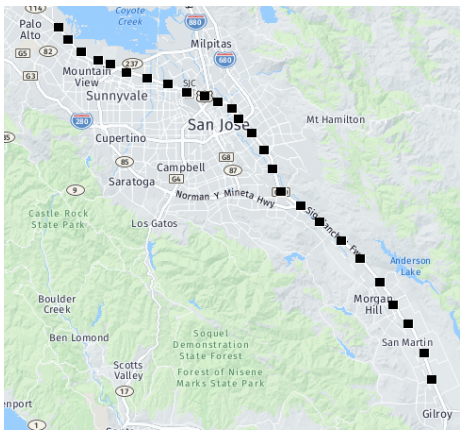


Figure 3.1: 26 sensors on one highway in Bay Area, California are selected. The black boxes are the main-line loop detector sensors.

3.4.2 Temporal clusters

The latent feature representation of one sensor's traffic states are represented in Fig 3.2. A t-distributed stochastic neighbor embedding (TSNE) introduced in [84], method is used for showing latent features in two dimension space. The color of each traffic state represents the hours of a day. One day is grouped into 10 colors, every 2.4 hours. The plot is for five weekdays. The results show that different time stamps are distinguishable from each other, e.g., all of the data points around 3pm are in same area. While the latent feature representation shows that there is a temporal pattern in data, we need to examine the similarity of time series based on latent feature representation. In Fig 3.3a, we select three areas and two data points from each area. The corresponding traffic flow is represented in Fig. 3.3b, 3.3c and 3.3d. To simplify the figures, we only present traffic flow, and ignore traffic occupancy and speed. The similarity between green and red lines represents the non-linear similarity between two closest data points. The figures clearly illustrate the meaningful relation among time series data and their corresponding latent features. In other words, the slope and non-linear temporal relation exists in the samples. Also, we note that

as it is illustrated in Section 4.1 (Data), all of the data points are subtracted from their first element to make them stationary, hence the first element is zero in all of the plots.

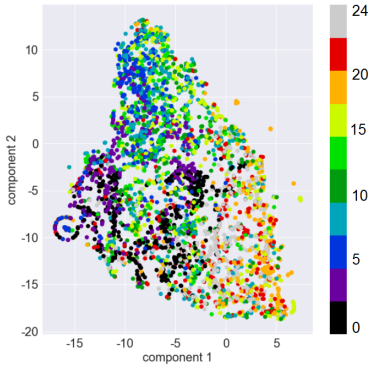


Figure 3.2: TSNE representation of autoencoder's latent features.

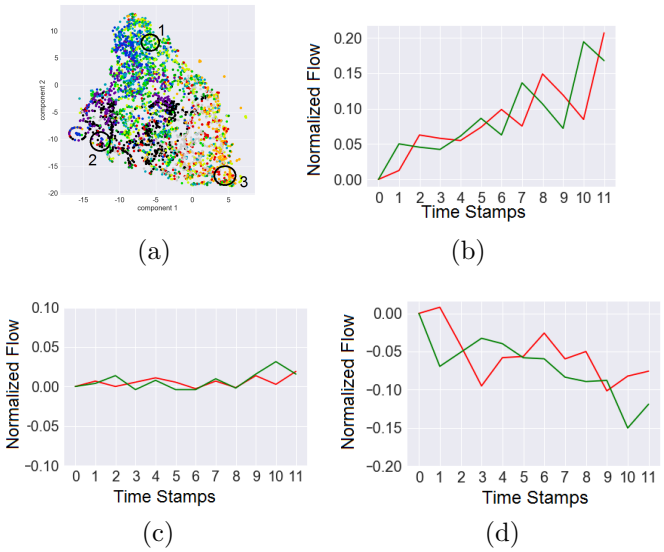


Figure 3.3: To represent the relation between latent features and time series data points, two data points are selected from three regions 1, 2 and 3 in Fig.a and represented in Fig.b, Fig.c and Fig.d.

Another interesting insight in temporal clusters is to find the relation among the clusters, and their DTW distance. In Fig 3.4, the relation between DTW and latent features is represented. For any given data points, the euclidean distance of latent features is calculated. Also the DTW among their time series is calculated. The correlation between euclidean distance of latent features and DTW of time series is presented in the Figure 3.4. Such an analysis illustrates that latent feature representation has a direct relation with DTW. The correlation

Figure 3.4: The plot for the relation of DTW distance and latent feature space.

between latent features and dynamic time warping is obtained and illustrated in Fig 3.4. The figure shows the changes of DTW for various sizes of latent features. Such an analysis shows that with a latent feature size of 4, the correlation between dynamic time warping and euclidean distance of latent features is 0.98. Such a high value of correlation shows the direct relation between them. A K-means clustering algorithm has a $O(n^2)$ computational time. Using a DTW method on time windows of size w, the total computation would be $O(n^2w^2)$. On the other hand, euclidean distance of two vectors with size of l has l comparisons. Then, computational time of K-means on latent feature space with euclidean distance would be $O(n^2l)$, where latent feature size l is very smaller than original size of time series w. This reduction in computational time of k-means clustering and high correlation between euclidean distance of latent feature representations and dynamic time warping is our first conclusion on the dataset.

The analysis in this section illustrates that latent features of traffic states keeps the non-linear relation among temporal data. A deep embedded clustering model uses the clustering of latent features. In next, we illustrate the patterns in output of deep embedded clustering model.

Deep embedded clustering uses latent feature representation of time series to find their clusters. The output of clustering layer is a vector $\mathbf{c} \in \mathbb{R}^K$, where $\mathbf{c}_k$ is the probability that

a data point assigned to the cluster k. To analyze the patterns, we use tsne for representing the vector $\mathbf{c}$ over different hours in Fig 3.5. The results show that different hours have their own clustering patterns and the cluster assignment are separable based on time stamp. This figure shows that not only latent features have distinguishable temporal patterns, but also cluster probabilities are distinguishable over different time stamps.

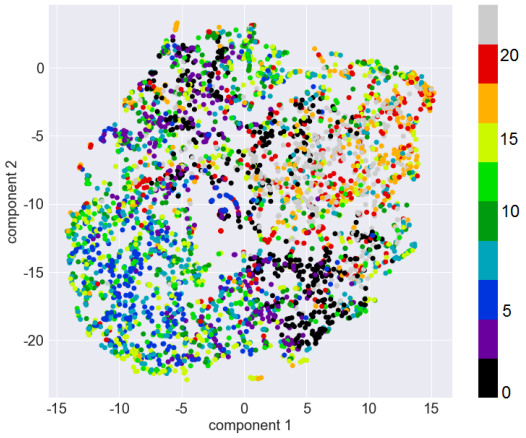


Figure 3.5: TSNE of cluster probabilities as the output of deep embedded clustering.

In clustering algorithms, finding an appropriate number of clusters is a challenging task. In spatio-temporal data, data have both context of time and space. We cannot easily estimate the number of clusters for temporal data. In deep embedded clustering, the number of initial clusters should be given. To obtain an appropriate number of clusters in k-means clustering, a general approach is to use inertia, as the sum of squares of data points to their cluster centers. In Fig 3.6, the plot shows the value of inertia for the given number of clusters. The results show that the optimum number of clusters is around 70. The optimum value can be obtained, when the reduction in inertia becomes linear.

The histogram of the size of clusters is represented in Fig 3.8. The histogram shows the size of clusters obtain by deep embedded clustering. There are some clusters with very small size. These clusters can contain rare time series shapes. On the other hand, the large clusters contain the most frequent patterns.

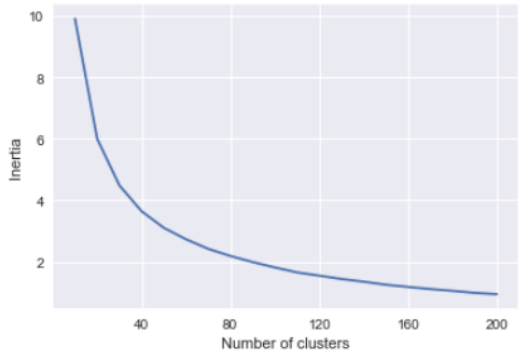


Figure 3.6: The plot of sum of squares of data points to cluster centers in terms of the number of clusters.

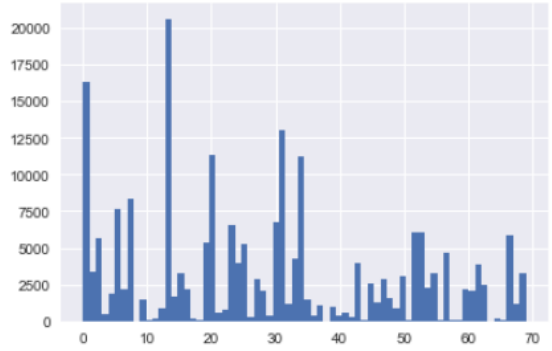


Figure 3.7: The histogram of size of temporal clusters.

To illustrate similarity of data points in each cluster, two data points are selected from cluster 20 and 30. In Fig. 3.9, it shows the similarity of selected data points in same clusters. Each cluster contains the most similar time series data.

Here we evaluate temporal clusters by comparing temporal similarities between members. A better clustering model should find clusters that have lower distance between members. Hence, we compare DEC and k-means clustering. The comparison is based on DTW distance function in Figure .

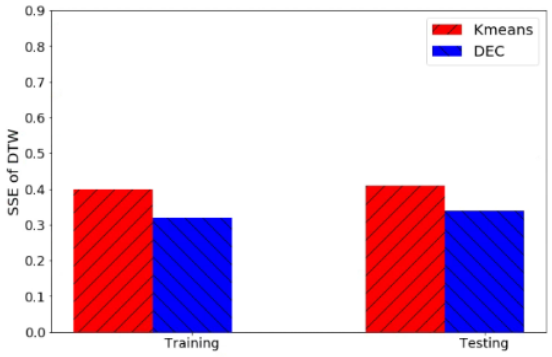


Figure 3.8: Sum of Square Error of DTWs

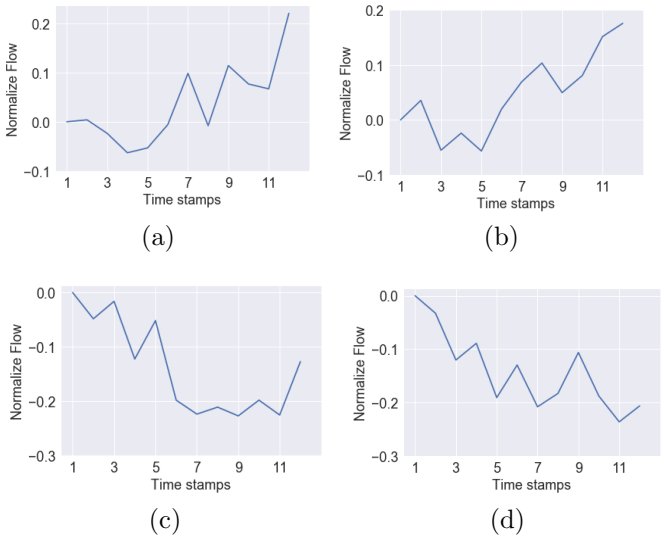
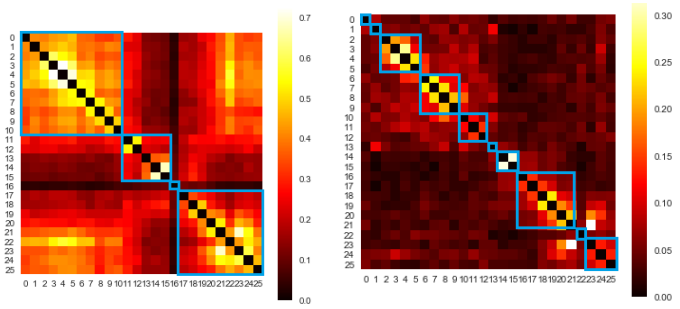


Figure 3.9: Four sampled data points are selected. The Fig.a and Fig.b show the data points in cluster 20. Fig.c and Fig.d show the data points in cluster 30.

3.4.3 Spatial clusters

In the previous section, we thoroughly analyze the temporal clusters. The analysis shows that there is clearly an interesting relation between latent feature clustering and dynamic time warping distance. Also, it shows that there is meaningful temporal patterns in the clusters. These analyses are the primary work for finding spatial clusters, because in Algorithm 1, we use temporal clusters for finding spatial clusters. In this section, we analyze the spatial clusters obtained by Algorithm 1.



(a) Similarity of sensors in one highway in off-peak hours,3AM.

(b) Similarity of sensors in one highway in peak hours, 5pm.

Figure 3.10: Heatmap of similarity matrix of sensors and the spatial clusters represented with blue rectangles.

We consider 70 clusters in deep embedded clustering. Each sensor i has $\bar{\mathsf{t}}$ data points, obtained by sliding window approach. The number of clusters to which each sensor is assigned has a mean of 40.4 and standard deviation of 6.5. In other words, the data points of each sensor are only assigned to 57% of the clusters on average. The analysis shows that each sensor is only assigned to part of the clusters, and it shows the output of clustering is different over spatial areas.

Algorithm 1 finds spatial clusters by comparing temporal clusters. The output of algorithm is a similarity matrix $\mathbf{C}^{\mathsf{T}}$ for a time interval $\mathbf{T}$. In the first analysis we divided traffic data into two time intervals. Figure 3.10 shows the similarity matrix with a heat-map. The result is presented for off-peak hours and peak hours. The output shows that in off-peak hours sensors are highly similar in all areas. However, in peak hours, only neighboring sensors are similar to each other. In other words, most part of the highway have similar flow patterns in off-peak hours. However, traffic congestion propagates flow in neighboring areas in peak hours. It increases dissimilarity between far sensors. This property shows that the number of clusters reduces in off-peak hours and increases in peak hours. To evaluate this property, we find the average value of similarity matrix in different hours of day in Fig 3.11. The results show that in peak hours there is more dissimilarity among far distance sensors and as a result the size of clusters are reduced.

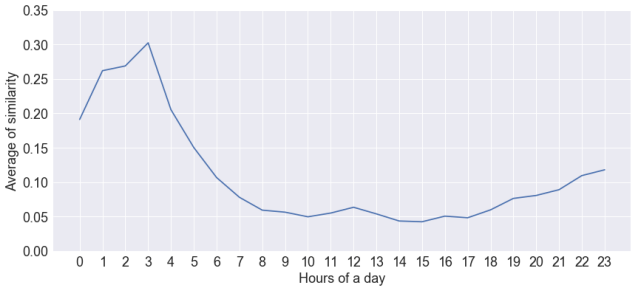


Figure 3.11: The relation between average similarity matrix and the hours of a day.

A threshold on similarity matrix finds the spatial cluster of neighboring similar sensors. The blue rectangles in Fig 3.10 represents the spatial clusters. In a similarity matrix, if higher similarities occurs near to diagonal, it shows that neighboring sensors are more similar to each other. The results show that in off-peak hours, there is more similarity in larger areas. In total, there are 4 clusters. However, in peak hours, there are 10 clusters. The neighboring sensors have more similar temporal clusters. This can be the result of flow propagation in the network, which results in higher similarity in neighboring sensors. The number of spatial clusters obtained by the proposed method over different hours of a day is in Fig 3.12. Higher number of clusters in peak hours shows that there is more similarities in a geographical area, when there is more flow congestion. Another interesting result is that the spatial clusters are not distributed. It shows that there is a meaningful relation between spatial similarity and distance. Also, our analysis shows that if we obtain temporal clusters with a k-means clustering model, then there is not any meaningful relation between neighboring

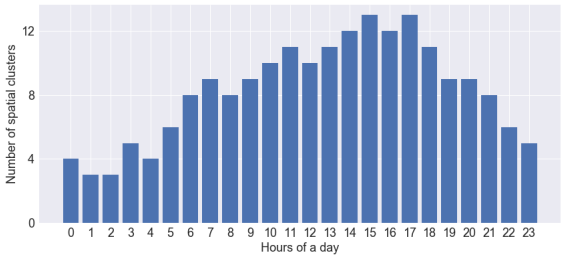


Figure 3.12: The number of spatial clusters for each hours of day is shown.

Further investigation can extract more insights related to spatial clustering of traffic flow.

For most of the hours of the day the pattern in the highway is the same as Fig 3.13. In large number of time intervals in peak hours, the sensors number 16 and 22 are distinguishable from the neighboring sensors. They have their own cluster. While we only consider mainline sensors in dataset, we found both of the sensors have two off-ramps and on-ramps before and after. It shows that their traffic patterns are affected by traffic flow comes from the outside of the highway, and they are not similar to neighboring mainline sensors. Such an analysis can extract interesting relation among highways and incoming/outgoing flows.

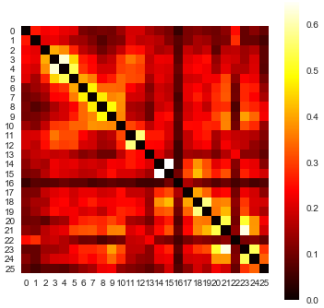


Figure 3.13: Heatmap of similarity of sensors. Existence of locality in traffic flow data. Similarity matrix of sensors.

3.5 Conclusion and future work

While clustering of spatio-temporal data is a challenging and important problem, there is a lack of analysis on applying deep embedded clustering models on spatio-temporal data to investigate their corresponding patterns. In this chapter, we formulate spatio-temporal clustering, define spatial and temporal clusters, and describe an approach in finding such clusters. The results of this work is presented in [15]. To find a more efficient way of clustering, we use a deep embedded clustering model, which clusters latent feature representation of time series data. A deep analysis of temporal and spatial clusters show that they preserve temporal and spatial patterns. We describe several interesting patterns extracted from the clusters, such as high correlation between euclidean distance of latent features and DTW distance of time series, distinguishable clustering probabilities for different time stamps, and

dynamic spatial clustering for various hours of a day. Unlike most of the application of time series clustering, here a dynamic spatial clustering can be more effective, when use such clusters for other analysis or machine learning models.

The proposed approach finds temporal and spatial clusters for spatio-temporal problems. There are some advantages of applying deep embedded clustering method compared with k-means clustering method. First, we describe that the computational time of deep embedded clustering is lower than k-means clustering. Also, while applying k-means clustering method on temporal data requires to define a non-linear distance function, a deep embedded clustering method do not require to have a non-linear distance function, since latent feature representations have a direct correlation with DTW method. Moreover, the deep embedded clustering method has a probabilistic output, which can be useful when the objective is to find fuzzy clusters.

This clustering approach can improve the performance of machine learning models in various problems, such as missing data imputation, forecasting and anomaly detection problems. For example, spatial clustering can find the most similar sensors in a region, and can be used for imputing missing data, or the distance of data points from temporal cluster centers can be used to detect anomalies. Moreover, the clustering of traffic data can help researchers to have further pattern analysis. Other types of neural networks, such as convolutional-recurrent neural, networks for finding an appropriate latent feature representation and other clustering models, such as agglomerative clustering model, can be investigated to better represent the clustering of spatio-temporal data. Also, this approach can find spatial and temporal clusters for other spatio-temporal data, such as weather data and power grid network data.

Chapter 4

Spatio-temporal missing data imputation

When sensors collect spatio-temporal data in a large geographical area, the problem of missing data cannot be escaped. Missing data negatively impacts the performance of data analysis and machine learning algorithms. In this chapter, we study deep autoencoders for missing data imputation in spatio-temporal problems. We propose a convolution bidirectional-LSTM for capturing spatial and temporal patterns. Moreover, we analyze an autoencoder's latent feature representation in spatio-temporal data and illustrate its performance for missing data imputation. Traffic flow data are used for evaluation of our models. The result shows that the proposed convolution recurrent neural network outperforms state-of-the-art methods. The results of the work is published in [14].

4.1 Introduction

Spatio-temporal problems have been studied in broad domains [17], such as transportation systems, power grid networks and weather forecasting, where data is collected in a geographical area over time. Traffic flow data are an important spatial-temporal data. Unlike traditional methods for static network flow problems [12], in which solving an optimization problem finds the solution, recently data-driven spatio-temporal approaches have been broadly applied on traffic flow data [20]. Spatio-temporal data are gathered by a large number of sensors and they inevitably miss observations due to a variety of reasons, such as an error prone measurements, malfunctioning sensors, or communication error [73]. In the presence of missing data, the performance of machine learning tasks such as classification, clustering and forecasting drops dramatically and results in biased inference. Hence, the problem is addressed by estimating missing values or by developing robust machine learning techniques with respect to missing data.

Statistical and machine learning techniques are broadly applied for missing data imputation. The primary approach is to use an ARIMA model, which works well under linear assumptions [9]. A matrix completion method has also been proposed for missing data imputation [141]; however, it requires low-rankness and static data. Dimensional reduction techniques for missing data imputation have good performance, e.g., a probabilistic principle component analysis method for missing traffic flow data [103], and a tensor-based model for traffic data completion [105]. Most recently, [71] proposes a clustering approach in spatial and temporal contexts for missing data imputation, including pattern clustering-classification and an Extreme Learning Machine with in-depth review of related work of missing data imputation in traffic flow problems. While clustering and dimensional reduction techniques differ from our model, some similarities suggests an avenue for further investigation in the future.

Increasing in the size of spatio-temporal datasets motivates researchers to develop scalable missing data imputation techniques, which is an motivation for developing new frameworks for big data analytics [124]. Contrary to statistical and traditional machine learning techniques, neural networks do not rely on hand-crafted feature engineering and do not use prior assumptions on input data. Shallow neural networks are shown to have great performance compared with other machine learning algorithms on traffic data [10], but their performance reduces in large-scale problems. Recently deep neural networks significantly improve performance of machine learning tasks on large-scale problems. Following the proposed denoising autoencoder with a fully connected neural network in [127], a comparison of denoising autoencoders and k-means clustering for traffic flow imputation is studied in [43]. Multiple missing data imputation with multiple training of fully connected, overcomplete autoencoders are examined in [46]. In [37], in-depth comparison of stacked autoencoders with state-of-the-art techniques for missing data imputation is studied.

Since training neural networks is computationally expensive, fully-connected, multiply trained and overcomplete autoencoders can be inefficient solutions for large scale problems. Moreover, recent works demonstrate the increased performance of convolutional layers and LSTM layers for extracting spatial and temporal patterns compared to fully connected layers. A Convolutional neural network is proposed for missing data imputation in traffic flow data [152]. The model captures spatial and short term patterns with a convolutional layer. A bidirectional LSTM with a modification on the LSTM neurons is proposed [30], but spatial relation is not considered. Convolutional recurrent neural networks have great performance in large-scale spatio-temporal problems [16]. In [55], a spatio-temporal autoencoder is proposed for high dimension patient data with missing values. However, their objective is to learn data with missing values for a classification problem.

In the aforementioned works deep neural networks have been studied on spatio-temporal data. However, there is lack of analysis in applying convolutional-recurrent autoencoders on

spatio-temporal problems for missing data imputation in traffic flow problems with the objective of learning spatial patterns with convolutional and temporal patterns with LSTM layers. In this paper, we first propose a convolutional recurrent autoencoder for multiple missing data imputation. The model is examined on traffic flow data. It is shown that the proposed convolutional recurrent autoencoder improves the performance of missing data imputation problem. Moreover, the latent feature representation of the autoencoders is analyzed. This analysis shows that the latent feature space is semantically meaningful representation of traffic flow data. We also examine the performance of applying k-nearest-neighbor (KNN) to evaluate the effectiveness of using autoencoders' latent representation in missing data imputation. The proposed model can be applied for missing data imputation in spatio-temporal problems.

4.2 Preliminaries

4.2.1 Problem definition

Spatio-temporal data is represented by a matrix $\mathbf{X} \in \mathbb{R}^{\mathsf{s}\times\bar{\mathsf{t}}\times\mathsf{f}}$, where s is the number of sensors, $\bar{\mathsf{t}}$ is the number of time steps and f is the number of features. Missing data can exist in various ways, for example at individual points or over intervals, where one sensor loses data for a period of time. To apply a deep neural network for time series imputation, a sliding window method generates $\mathbf{x}^{\mathsf{t}} \in \mathbb{R}^{\mathsf{s}\times\mathsf{w}\times\mathsf{f}}$, where w is time window and $\mathsf{t} \in [0, \bar{\mathsf{t}}]$. In the rest of the paper, we call $\mathbf{x}^{\mathsf{t}}$ as a data point. For the purpose of training and evaluation, an interval of missing values is added to the input data and represented with $\mathbf{x}^{\mathsf{t}}_{\mathsf{m}}$. The objective is to impute missing values for $\mathbf{x}^{\mathsf{t}}_{\mathsf{m}}$ using spatial and temporal correlation. In Fig. 4.1, a schematic example of applying a sliding window on a spatial time series with interval-wise missing values is represented.

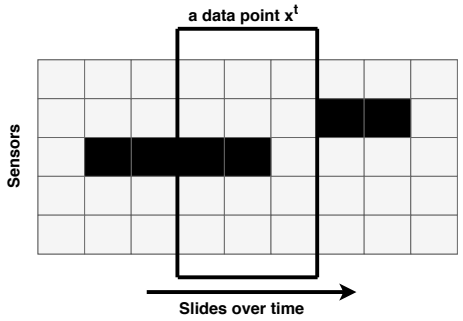


Figure 4.1: A sliding window selects subsamples and feeds these into an autoencoder. The missing values are represented in black.

4.2.2 A denoising autoencoder

An autoencoder decoder $\mathsf{AD}(.)$ proposed in [127] and can be applied in missing data imputation problem. In the training process, a denoising encoder decoder receives $\mathbf{x}_{\mathsf{m}}^{\mathbf{t}}$ as input and $\mathbf{x}^{\mathbf{t}}$ as target data. It reconstructs its input $\bar{\mathbf{x}}^{\mathbf{t}} = \mathsf{AD}(\mathbf{x}_{\mathsf{m}}^{\mathsf{t}})$ by minimizing the loss function $\mathsf{Loss}(\mathbf{x}^{\mathsf{t}}, \bar{\mathbf{x}}^{\mathsf{t}})$, e.g. mean square loss function, for autoencoders' output $\bar{\mathbf{x}}^{\mathbf{t}}$. In other words, the autoencoder receives a data point with some missing values and reconstructs it with the objective of accurate missing data imputation. An encoder reduces the dimension to a latent feature space $\mathbf{h} \in \mathbb{R}^{\mathsf{d}}$, where $\mathsf{d} < \mathsf{n}$, which extracts the most important patterns of the input data. The decoder reconstructs the input from its latent representation. For a two layer encoder decoder, an encoder is represented with $\mathbf{h} = \sigma(\mathsf{drop}(\mathbf{x})\mathbf{w}^1 + \mathbf{b}^1)$ and a decoder is represented with $\bar{\mathbf{x}} = \sigma(\mathsf{drop}(\mathbf{h})\mathbf{w}^2 + \mathbf{b}^2)$, where $\sigma(.)$ is the activation function and $\mathsf{drop}(.)$ is dropout function. A multi layer fully connected, convolutional or recurrent neural network can be used as an encoder or decoder to reconstruct input data. Moreover, learning low dimensional representation of spatial data is shown effective in unsupervised learning [50] and autoencoders broadly have been used to produce latent feature representation on real-world datasets [72]. In section 4.3.2, we describe the effectiveness of using latent feature representation in multiple missing data imputation problem.

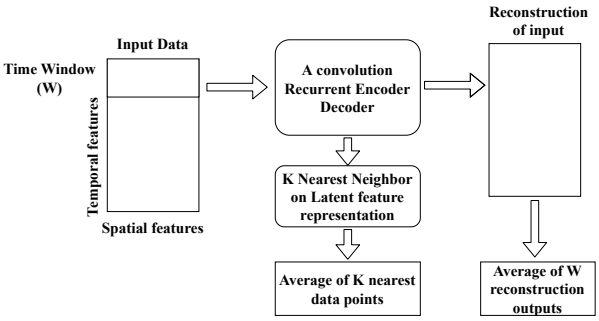


Figure 4.2: The framework for multiple imputation with autoencoders

4.3 A convolutional-recurrent deep neural network encoder decoder framework

As discussed in detail in [111], in multiple imputation each missing datum is replaced with weighted average of more than one imputations. Hence, we propose a framework for multiple missing data imputation on spatio-temporal data, represented in Fig. 4.2.

A sliding window method gives the input data with size of w to the autoencoder. A convolutional recurrent autoencoder reconstructs the input data and automatically imputes missing values. There are w reconstructed values for each time window. The average of these reconstructed values is the output of neural network. The evaluation of reconstructed values is shown in Section. 4.4.5. The second approach is with the latent feature representation of autoencoders. A KNN finds the most similar k data points in training data. The average of these produces the imputed values for the testing data. The model is evaluated in Section. 4.4.7.

4.3.1 A CNN-BiLSTM autoencoder

Here we introduce the proposed convolutional recurrent autoencoder for spatio-temporal missing data imputation. The proposed model is illustrated in Fig. 4.3.

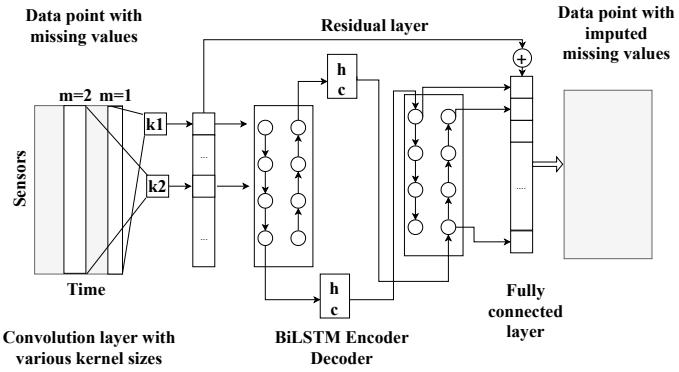


Figure 4.3: A convolutional BiLSTM encoder decoder (CNN-BiLSTM-Res) for missing data imputation

To extract spatial and temporal patterns, an encoder consists of both convolutional and LSTM layers. A convolutional layer has a kernel, which slides over spatial time series data $\mathbf{X} \in \mathbb{R}^{s \times w \times c}$, where c is the number of channels. For non-grid data, sliding a kernel on spatial features loses the network structure and reduces the performance of model [16]. Hence the kernel only slides over the time axis. The kernel size is (s, m), where $m < w$, and stride size is $(s, 1)$. Various length of kernel have been shown to have better performance. Hence, several kernels with different values of m are applied to the input data. The output of each kernel i is $\mathbf{k}^i \in \mathbb{R}^{1 \times w \times f}$, where f is the filter size. All of the outputs are concatenated and represented with $\mathbf{h} \in \mathbb{R}^{1 \times w \times F}$, where F is the total size of all filters.

An LSTM layer receives the output of convolutional layer, represented by $\mathbf{X} \in \mathbb{R}^{w \times F}$. An LSTM cell uses input, output and forget gates to prevent vanishing gradients in recurrent cells. It also returns hidden state $\mathbf{h}^t$ and cell state $\mathbf{c}^t$. A bidirectional LSTM layer captures the relation of past and future data simultaneously. It has two sets of LSTM cells which propagate states in two opposite directions. Thus a bidirectional LSTM layer is used for the recurrent component. Given l_1 as the number of units in LSTM layer, the output of bidirectional LSTM is $\mathbf{h} \in \mathbb{R}^{w \times 2 \times l_1}$. The latent feature representation of encoder consists of LSTM states $[\mathbf{h}^t_{\text{forward}}, \mathbf{c}^t_{\text{forward}}, \mathbf{h}^t_{\text{back}}, \mathbf{c}^t_{\text{back}}]$, where these are the hidden and cell states of the forward and backward direction of bidirectional LSTM.

The decoder receives the encoder states and encoder output. The decoder consists of a bidirectional LSTM and a fully connected layer. The LSTM layer receives the hidden and cell states of the encoder to reconstruct the input data. A bidirectional model reconstructs past and future data. It follows with a fully connected layer with linear activation function.

Training the encoder decoder with convolutional and LSTM layers is slow, as the gradient of the loss function is propagated backward on to LSTM cells and then convolutional layers. To increase the speed of training, we used a skip connection, introduced in [51], to connect the output of the convolutional layer to the fully connected layer with a $\mathsf{Add}(.)$ function. In the training process, the convolutional layer receives more effect from the gradient of loss function and as a result, there is faster convergence for the encoder decoder to learn spatial and temporal patterns.

The reconstruction of input automatically imputes missing data from the spatial and temporal correlation among neighboring areas. Given a time window w, every time stamp $\mathbf{x}^{\mathsf{t}}$ is reconstructed w times and the average is used for missing imputation. An autoencoder decoder reconstructs input data $\bar{\mathbf{x}}^{\mathsf{t}} = \mathsf{AD}(\mathbf{x}^{\mathsf{t}}_{\mathsf{m}})$ by minimizing loss function $\mathsf{Loss}(\bar{\mathbf{x}}^{\mathsf{t}}, \mathbf{x}^{\mathsf{t}})$ for all time steps t.

4.3.2 Missing data imputation using latent feature representations

A KNN algorithm compares the distance among all data points, and finds the k nearest data points. This approach find the most similar data points and then find the average for missing data imputation. With a sliding window approach, the number of data points in training data is the same as number of time steps $\bar{\mathsf{t}}$. For a given data point $\mathbf{x}^{\mathsf{t}}$ which is a matrix of size $\mathsf{len} = \mathsf{s} \times \mathsf{w} \times \mathsf{f}$, the total number of comparison in KNN is $\mathsf{t}^2 \times \mathsf{len}$. Moreover, a time series distance can be obtained with Dynamic Time Warping [110], which is computationally more expensive than euclidean distance.

The latent representation of autoencoder is a fixed size and reduced dimension vector $\mathbf{h} \in \mathbb{R}^{\mathsf{d}}$. Applying KNN on latent representation is computationally more efficient than on time series data points. The total comparison is $\mathsf{t}^2 \times \mathsf{d}$ and the latent feature distance can be computed with euclidean distance, faster than Dynamic Time Warping. In the experimental analysis, we evaluate the computational time of applying KNN on latent feature. Moreover, the average of k most similar data points is used as multiple missing imputation. The results of this analysis is compared with PCA-KNN in experimental results.

4.4 Experimental results

4.4.1 Dataset

We examine the performance of the proposed model on traffic flow data available in PeMS [1]. Traffic data are gathered every 30 seconds and aggregated every 5 minutes using loop detector stations on highways. We use three subset of stations in the Bay Area, represented in Fig. 4.4, and evaluate the average performance of our model on these three regions to have better evaluation of the models. The first region has 10 mainline stations on 22 miles of highway US 101-South. The second region has 9 mainline stations on 10 miles of I-280-South highway, and the third region has 11 mainline stations on 13 miles of I-880-South. The training data is for the first 4 months of 2016 and the testing data is for next 2 months. The selected sensors have more than 99% of available data for this time period.

4.4.2 Preprocessing

The data is scaled to range of [0-1] where for each data set, 0 is the minimum flow observed and 1 is the maximum. A sliding window approach is used to generate image-like input for

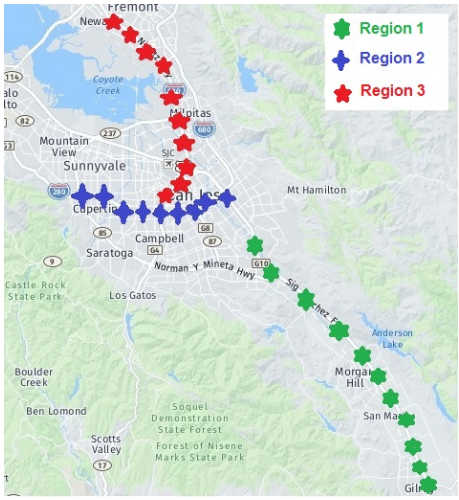


Figure 4.4: Three regions of highways are selected for missing data imputation analysis.

time series data. During the experiments, we found out a time window of size 6, 30 minutes, works well. Each data point is represented with $\mathbf{X}^t \in \mathbb{R}^{s \times 6 \times 1}$, where s is the number of sensors for each region.

To evaluate the model for missing data imputation, we added missing blocks to have a ground truth for evaluation. The missing data is generated randomly on training and testing data. We generated blocks of missing data with size of 0.5 to 4 hours. The sensors are randomly selected for each missing block. In the analysis, training data without missing values cannot result in a robust autoencoder for missing data imputation. Therefor, 25% percent of training and testing data is considered as missing values. In the analysis, the performance of missing data imputation models are examined only on these missing blocks, represented with index list of $\mathbf{I}_m^{test}$.

4.4.3 Baseline missing data imputation models

Our first missing data imputation method uses a temporal average to fill missing data. Traffic flow patterns are repeated every week. Hence, a weekly-hourly average table is obtained from training data (W-H-Average). The main drawback of using temporal average is that specific

days such as holidays or event days (games, festivals, concerts) have their own patterns and they are not repeated in the training data.

The second method uses the closest sensors to estimate the missing data. The value of traffic flow should be similar to the closest sensors on highways. Following the work [16], a Dynamic Time Warping distance method finds the most similar sensors using time series residuals. The method uses the average of the two closest sensors and estimates the missing data (Neighbor-Value).

In the third baseline method, the most important principle components are selected, then a KNN finds the most similar data points. The average of k nearest values is used to estimate missing data (KNN-PCA). In the analysis, we examine different values of PCA components. The first 10 components contain more than 95 % information ratio. Also, larger values of k, improves the result, as the average of several missing imputations is usually a better estimation for missing values. The best size of PCA components and k are 10 and 20, respectively. The number of features is the number of sensors multiplied by time window, which is 60, 54 and 66 for three regions. The best values of MAE and RMSE are shown in Table. 4.1.

4.4.4 Autoencoder models

Here we describe the implemented autoencoders. For all of the models, the batch size is set to 256 and the epochs are set to 100. An ADAM optimizer with learning rate of 0.001 is used for training the model.

A fully connected denoising encoder decoder is implemented for missing imputation FC-NN. The model is trained with architecture of (32, 16, 12, 16, 32) obtained by grid search over various number of layers and hidden units. Each layer is a fully connected layer with a

Missing data imputation error for traffic flow data		
Models	MAE	RMSE
W-H-Average	26.3	34.8
Neighbor-value	38.9	45.5
KNN-PCA	19.0	25.5
FC-NN	14.3	21.5
LSTM	10.1	16.0
BiLSTM	7.8	14.0
CNN-BiLSTM	7.6	13.9
CNN-BiLSTM-Res	**6.8**	**13.0**

Table 4.1: The comparisons of the models

Leaky-RELU activation function.

To capture temporal patterns, an LSTM encoder decoder with 32 neurons is trained LSTM. To capture the effect of past and future data points, a bidirectional LSTM is implemented with 16 neurons in each direction BiLSTM. A dropout with parameter 0.2 prevents overfitting the LSTM layers. A convolutional recurrent encoder decoder CNN-BiLSTM is implemented with four kernels of size $(\mathsf{s}, 1)$, $(\mathsf{s}, 2)$, $(\mathsf{s}, 3)$ and $(\mathsf{s}, 4)$ and filter size of 8 and a Leaky Relu activation function. The bidirectional-LSTM has 16 units on each direction and is connected to a fully connected layer with the size of input sensors. Slow convergence of convolutional-BiLSTM model motivates us to add a skip connection connecting convolutional to the output of BiLSTM for faster gradient propagation. The model CNN-BiLSTM-Res, the proposed model in Fig. 4.3, is with the same architecture of CNN-BiLSTM but with skip connection. All implementations has been done with Tensorflow and Keras [2].

4.4.5 Comparison of results

Given $\mathbf{x}$ as real value and $\bar{\mathbf{x}}$ as predicted value, Mean Absolute Error (MAE), and Root Mean Square Error (RMSE) are used for evaluation. Given a set of missing data points in testing data $\mathbf{X}_{\mathsf{m}}^{\mathsf{test}}$ and their corresponding indices $\mathbf{I}_{\mathsf{m}}^{\mathsf{test}}$, the index i is selected from the index set of

missing data $\mathbf{I}_{m}^{test}$ in 4.1 and 4.2.

$$MAE = \frac{1}{n} \sum_{i \in \mathbf{I}_{m}^{test}} |\mathbf{x}_i - \bar{\mathbf{x}}_i| \tag{4.1}$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i \in \mathbf{I}_{m}^{test}} (\mathbf{x}_i - \bar{\mathbf{x}}_i)^2} \tag{4.2}$$

The results are represented in Table. 4.1. It shows that the temporal and spatial averages, the first two models have a poor performance for missing data imputation. Among three baseline models, KNN-PCA is the best missing data imputation technique. Autoencoders have significantly better performance than baseline models. The LSTM model has good performance for missing data imputation compared with FC-NN for capturing temporal patterns. A bidirectional LSTM shows great performance by capturing the relation between past and future data simultaneously. A CNN-BiLSTM hardly converges to the optimum solution but is not better than the BiLSTM model. Finally, the proposed CNN-BiLSTM-Res encoder decoder has the best MAE and RMSE. It shows that a skip connection improves the performance for a combination of convolutional and LSTM layers. The model CNN-BiLSTM-Res has 13% and 7% improvement on MAE and RMSE compared with the best BiLSTM model. As it is illustrated in Section. 4.3.1, because of the slow convergence of convolutional LSTM models, a skip connection is used to propagate gradients of loss function directly to convolutional layer. In Fig. 4.5, the convergence of CNN-BiLSTM and CNN-BiLSTM-Res are represented, which shows faster convergence of CNN-BiLSTM-Res.

In Fig. 4.6, the prediction results is represented for FC-NN and CNN-BiLSTM-Res as the example of missing data imputation results. Compared with FC-NN, the prediction result of CNN-BiLSTM-Res is clearly more accurate missing imputation and closer to ground truth. In Fig. 4.7, the plot illustrates the missing data imputation by CNN-BiLSTM-Res for two

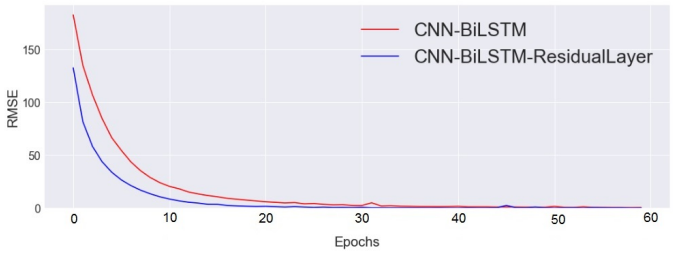


Figure 4.5: The comparison of validation loss during training of autoencoder models

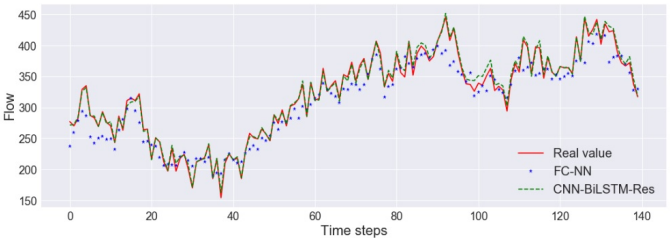


Figure 4.6: The comparison of missing data imputation models for one interval of missing values

missing blocks during three days, and shows the closeness of imputed data to real traffic flow data. This output example shows the estimation of missing block of data is very close to real values; however, still the distance between real and predicted values for missing blocks is more than healthy data, which are the time series values out of missing blocks.

4.4.6 Discussion on multiple missing data imputation

For non-temporal data, an autoencoder reconstructs one value for each input data point. However, for temporal data, a sliding window generates data points for each time step. Referring to Figure 4.1, the data point actually contains all of the values within a time window. For a given time window w, there are w reconstructed values for each time step. The result in Table. 4.1 is for w multiple missing data imputation. Here we use one step reconstruction of each output for comparison purpose. In other words, here we describe a single missing imputation output of applying autoencoders on traffic flow data.

The value of MAE for FC-NN, LSTM, BiLSTM and CNN-BiLSTM are 23.7, 15.5, 11.9,

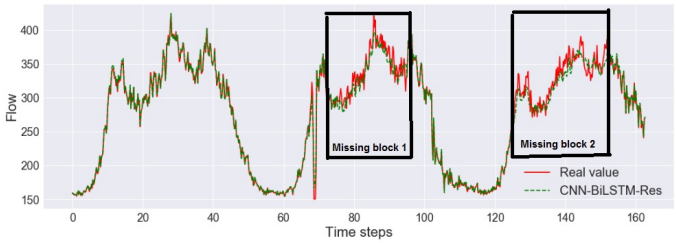


Figure 4.7: The illustration of missing data imputation for one sensor by the proposed model

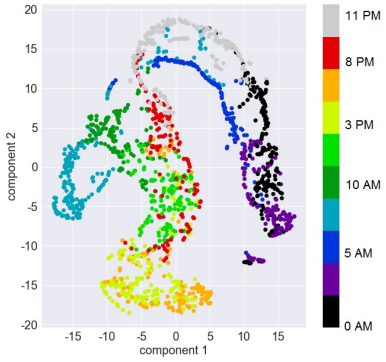


Figure 4.8: The latent feature space visualization of FC-NN with t-SNE. Each data point has a color that represents the time of day.

6.9, respectively. Also, the RMSE for FC-NN, LSTM, BiLSTM and CNN-BiLSTM are 32.1, 22.5, 18.1, 13.7, respectively. Comparing to Table. 4.1, we can see that a single missing data imputation has very lower performance. The analysis shows that multiple imputation and using the average of them significantly improves missing data imputation. This multiple imputation approach improves the output of autoencoders on time series data.

4.4.7 Latent feature representation

The latent feature representation of autoencoders illustrates meaningful information. In Fig 4.8, a t-SNE method [84] visualizes latent feature representation of hidden state of (FC-NN) for 7 days. The plot shows that for each time of day the patterns of data points are closer to each other. Here our objective is to illustrate how latent feature representation can be used for missing data imputation. Hence we use the concept of similarity of data points. A KNN

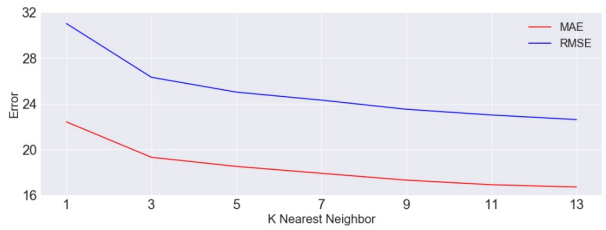


Figure 4.9: The comparison of applying KNN on FC-NN latent feature for various size of k

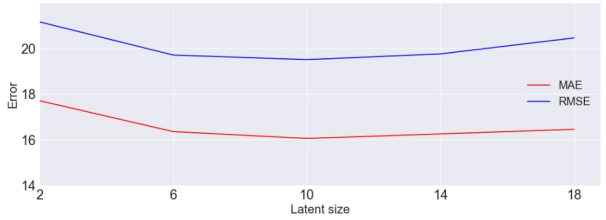


Figure 4.10: The comparison of applying KNN on FC-NN latent feature for various size of latent features

is applied on latent feature representation in training data points. The k most similar data points are used. The error for the average of different values of k is represented in Fig 4.9. The plot shows that a 1 nearest neighbor on latent feature representation results in 23.5 and 31.0 for MAE and RMSE scores. However, a 13 nearest neighbor results in 16.7 and 22.6, MAE and RMSE, respectively. The reduction in missing data imputation error shows the effectiveness of multiple imputation on latent feature representation. We also examine the relation between size of latent features and missing imputation on FC-NN in Fig. 4.10. The analysis shows that across latent sizes of 2 to 20 there are changes in the performance of the missing data imputation. The results suggest that the best latent size is 10.

A KNN is applied on latent feature representation of various implemented autoencoders. The results of applying KNN on latent feature of FC-NN, hidden and cell state of LSTM and BiLSTM have MAE of 16.6, 18.1, 17.8 and RMSE of 22.5, 24.1, 23.8, respectively. While a FC-NN with six layers is the best model to generate latent features, the other convolutional recurrent models cannot easily generates a latent feature representation for missing data imputation. One conclusion is that size of latent vector greatly effect on the result. A KNN on smaller size of latent vector finds better missing data imputation. The analysis also shows

that applying KNN on the latent feature of FC-NN is better than KNN-PCA, which shows autoencoders are capable of generating better latent feature representation for traffic flow data.

4.5 Conclusion and future work

In this paper, we study autoencoders for missing data imputation in spatio-temporal problems and examined the performance of various autoencoders to capture spatial and temporal patterns. We illustrate that a convolutional recurrent autoencoder can capture spatial and temporal patterns and outperforms state-of-the-art missing data imputation. We conclude that a convolutional layer with various kernel sizes and a bidirectional LSTM improves missing data imputation in traffic flow data. Also, the slow convergence of the convolutional recurrent autoencoder is improved with a skip connection. We also describe an approach considering multiple imputation for autoencoders for time series data. The results show that multiple imputation is significantly better than single imputation. Moreover, We illustrate advantage of using the latent feature of autoencoders for missing data imputation. We describe an approach for using autoencoders' latent feature representation for multiple imputation. The analysis shows that it outperforms KNN on principle components of traffic flow data. However, the latent feature of convolutional recurrent autoencoders needs a careful design of the architecture to obtain better results and can be explored more in future works.

Future research will focus on generative neural networks which recently shows a significant performance in various machine learning problems. Moreover, while it is shown that convolutional recurrent neural networks show a great performance for spatio-temporal problems, spatial and temporal clustering techniques can make the model more effective on larger geographical areas.

Chapter 5

Spatio-temporal forecasting

Spatio-temporal problems arise in a broad range of applications, such as climate science and transportation systems. These problems are challenging because of unique spatial, short-term and long-term patterns, as well as the curse of dimensionality. In this chapter, we propose a deep learning framework for spatio-temporal forecasting problems. We explicitly design the neural network architecture for capturing various types of spatial and temporal patterns, while the model is robust to missing data. In a preprocessing step, a time series decomposition method is applied to separately feed short-term, long-term and spatial patterns into different components of the neural network. A fuzzy clustering method finds clusters of neighboring time series residuals, as these contain short-term spatial patterns. The first component of the neural network consists of multi-kernel convolutional layers which are designed to extract short-term features from clusters of time series data. Each convolutional kernel receives a single cluster of input time series. The output of convolutional layers is concatenated by trends and followed by convolutional-LSTM layers to capture long-term spatial patterns. To have a robust forecasting model when faced with missing data, a pretrained denoising autoencoder reconstructs the model's output in a fine-tuning step. In experimental results, we evaluate the performance of the proposed model for the traffic flow prediction.

The results show that the proposed model outperforms baseline and state-of-the-art neural network models. The results of the work is published in [16].

5.1 Introduction and literature review

Time series data arise in broad areas, such as engineering, medicine, finance, and economics. Various types of statistical and machine learning techniques have been applied to such problems. Increases in both the volume and variety of data motivate researchers to develop scalable machine learning models for important time series problems, such as forecasting [81], anomaly detection [4], classification [149], and clustering [86]. Spatio-temporal data is a multi-variate time series data, in which there is a spatial dependency between neighboring time series [17]. Spatio-temporal forecasting problems arise in diverse areas, such as solar power forecasting [24], load demand forecasting [101], weather forecasting [137], various smart city applications [119], and transportation systems, such as the traffic flow prediction problem [100]. In this chapter, we propose a deep learning model for spatio-temporal problems, and focus on a short-term traffic flow prediction problem. The traffic flow prediction problem is a challenging problem, because of the spatial and non-linear relations among neighboring time series, as well as short-term and long-term temporal patterns.

Predicting traffic flow data can assist travelers to make better routing decisions, improve traffic management systems, and decrease traffic congestion. Recently, navigation systems available on smart phones have increased the importance of a traffic flow prediction problem in our daily lives. Many travelers rely on these to plan an efficient travel route. With the advent of new sensing, computing and networking technologies such as cameras, sensors, radars, inductive loops, and GPS devices, a large volumes of data are readily available [148]. These increasingly large data sets necessitate the development of scalable machine learning models [151] [35]. Hence, to improve the performance of transportation systems, researchers

are motivated to take advantage of new spatio-temporal data-driven techniques and to design scalable algorithms [6].

5.1.1 Background

Starting in the 1970's with the original work of Gazis and Knapp [45], many studies develop new models for traffic flow prediction problems, such as auto-regressive integrated moving average (ARIMA) [57] and Seasonal-ARIMA [68], and statistical techniques, such as Markov chain models [140] and Bayesian networks [129]. However, there are several limitations on these models, due to prior assumptions, lack of handling missing data, noisy data, outliers, as well as the curse of dimensionality. Neural networks, with fully-connected and radial basis layers, have been broadly applied to transportation problems [97, 10]. Several works show a great performance of neural networks in transportation problems for twenty-five years or more [116, 108, 58]. However, shallow architecture neural networks cannot efficiently work with large-scale data.

Deep learning models have been successfully applied to broad areas of machine learning problems. Recently, there have been several attempts to design deep learning models for time series forecasting problems. The primary work proposes a stacked autoencoder (SAE) model to learn traffic flow features and illustrate the advantage of SAE model versus multi-layer perceptron [81]. In [53], they propose stacked autoencoders with multi-task learning at the top layers of the neural network. A deep belief network (DBN) composed of layers of a restricted boltzman machine is proposed in [69]. In [130], an ensemble model of four categories of fully connected neural networks is proposed. In [102], an ensemble of DBN with Support Vector Regression is proposed. While most of these models use full-connected layers, other types of neural network layers can better capture spatio-temporal patterns.

Convolutional Neural Networks (CNN) have been applied to various data, such as images,

videos, and audios. Weight sharing, the main feature of convolutional layers, reduces the number of trainable variables and better captures locality in data. In [82], the performance of CNN models is examined for a time series forecasting problem, where spatio-temporal traffic flow data are represented as images. The CNN model forecasts traffic speed in a large transportation networks. In [40], they study image-like representation of spatio-temporal data using convolutional layers and an ensemble learning model. A convolutional layer considers spatial structure in euclidean space, which can miss the underlying structure of the network [52]. This problem is addressed in this chapter using a multi-kernel convolutional layer, applied to the clusters of similar time series data. As an alternative approach, following the work [28], spatial dependency is captured using bi-directional diffusion convolutional recurrent models [74]. They illustrate a graph-structured representation of time series data to capture spatial relation among time series.

Moreover, in the presence of temporal data, recurrent neural networks have shown a great performance in a time series forecasting problem. A Long-Short Term Model (LSTM) model [109] have been successfully applied to the time series forecasting problem [147], the traffic speed prediction problem [83] and traffic flow estimation with missing data [123] [13]. While convolutional neural networks can exhibit excellent performance on spatial data, and recurrent neural networks have advantages on temporal data; spatio-temporal problems combine both of these. [137] proposes a new convolutional-LSTM layer for the spatio-temporal forecasting problem, and applied to the weather forecasting problem. In [134], they propose a hybrid deep learning model for a traffic flow prediction. An LSTM component and a CNN component separately learn temporal and spatial features. A CNN model is proposed for the time series forecasting problem in [139]. They propose explicit grouping of input time series and implicit grouping using error back-propagation. In [32], they use a CNN-LSTM model to capture traffic flow evolution in a transportation network. A convolutional layer is followed by an LSTM layer, separately applied to downstream and upstream data. In [47], a 3-dimensional convolutional layer is applied to spatio-temporal traffic data. The model cap-

tures correlation of traffic data in both spatial and temporal dimensions. In [60], a structural recurrent layer is proposed, which converts topology of the road network into recurrent layers. In [75], they propose a model with convolutional and gated convolutional layers followed by attention layers. In [146], a graph convolutional sequence to sequence model is proposed for a multi-step prediction problem.

5.1.2 Contributions of the work

In the aforementioned works, spatio-temporal forecasting problems have been studied, and various types of convolutional and recurrent neural networks have been proposed. Spatio-temporal data contain several unique patterns which motivate us to use spatial and temporal decomposition methods, and to explicitly consider various types of patterns in designing an efficient neural network architecture. Here, we address some of the important challenges in designing a neural network architecture for spatio-temporal data. In these data, time series residuals are not meaningless noise, but they represent spatial patterns in data. Moreover, convolutional layers can capture spatial and short-term patterns, but sliding a convolutional kernel on spatial features miss the underlying structure of the network and reduce the performance of the model. Also, spatio-temporal data have a non-stationary behaviour in the existence of long-term patterns. Furthermore, spatial and temporal patterns can be used to handle missing data.

In this chapter, we consider unique properties of spatio-temporal data in designing a deep learning model, and we use spatio-temporal decomposition methods to extract spatial and temporal patterns. The contributions of the chapter are:

- We propose a deep neural network model, which explicitly consider various types of short-term, long-term and spatial patterns.

- We describe a fuzzy clustering method, which clusters spatial time series data.
- A multi-kernel convolutional layer is designed to maintain the network structure, and extract short-term and spatial patterns. It is followed by a convolution-LSTM component to capture long-term trends, and a pretrained denoising autoencoder to have robust prediction in the existence of missing data.
- The spatial and temporal patterns of a real world traffic flow data are analyzed, and the performance gains of the proposed model relative to baseline and state-of-the-art deep neural networks are illustrated.

5.2 Problem definition

Time series data are a set of successive measurements, $\mathbf{X}_{\mathsf{i}} = \{\mathbf{x}_{\mathsf{i}}^{1}, \ldots, \mathbf{x}_{\mathsf{i}}^{\bar{\mathsf{t}}}\}$, where $\mathbf{x}_{\mathsf{i}}^{\mathsf{t}}$ is observed values at location i and time stamp t, and the total number of time stamps is $\bar{\mathsf{t}}$. A sensor i gathers $\mathbf{x}_{\mathsf{i}}^{\mathsf{t}}$ with corresponding k features $\mathbf{x}_{\mathsf{i}}^{\mathsf{t}} = \{x_{\mathsf{i}}^{\mathsf{t},1}, \ldots, x_{\mathsf{i}}^{\mathsf{t},\mathsf{k}}\}$. A spatio-temporal data is a collection of n multi-variate time series $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{\mathsf{n}}\}$, represented by a matrix $\mathbf{X} \in \mathbb{R}^{\mathsf{n}\times\bar{\mathsf{t}}\times\mathsf{k}}$, where n is the number of sensors, which gather spatio-temporal data in a geographical area. In spatio-temporal data, there is a non-linear similarity and correlation among neighboring time series data.

Given $\mathbf{X}$ as the set of all time series in a region, a spatio-temporal forecasting problem is cast as a regression problem. The objective of a forecasting problem is to predict $\mathbf{X}_{\mathsf{output}}^{\mathsf{t}} = \{\mathbf{x}^{\mathsf{t}+1}, \ldots, \mathbf{x}^{\mathsf{t}+\mathsf{h}}\}$, given $\mathbf{X}_{\mathsf{input}}^{\mathsf{t}} = \{\mathbf{x}^{\mathsf{t}-\mathsf{w}}, \mathbf{x}^{\mathsf{t}-\mathsf{w}+1}, \ldots, \mathbf{x}^{\mathsf{t}}\}$, where w is the size of time window, and h is the prediction horizon. A sliding window method generates input data points $\mathbf{X}_{\mathsf{input}}^{\mathsf{t}}$ and target data points $\mathbf{X}_{\mathsf{output}}^{\mathsf{t}}$. In equation (5.1), an optimum parameter $\theta^{\star}$ represents the optimum time series forecasting model. In a neural network, $\theta^{\star}$ represents the weights of the model and the gradient descent algorithm minimizes the non-linear loss function $\mathsf{f}_{\theta}(.,.)$ by

solving following optimization problem,

$$\theta^{\star} = \text{argmin}_{\theta} \sum_{t=1}^{\bar{t}} f_{\theta}(\mathbf{X}^{t}_{\text{input}}, \mathbf{X}^{t}_{\text{output}}) \tag{5.1}$$

In this chapter, given a spatio-temporal data $\mathbf{X}$, a deep neural network receives an input data $\mathbf{X}_{\text{input}} \in \mathbb{R}^{n \times w \times k}$, and predicts $\mathbf{X}_{\text{output}} \in \mathbb{R}^{n \times h \times \bar{k}}$, where $\bar{k}$ is the number of output features.

5.3 Technical background

Here, we detail the main components of the proposed approach, including a fuzzy hierarchical agglomerative clustering, a convolutional layer, a convolutional-LSTM layer and a denoising autoencoder.

5.3.1 Dynamic time warping

A Dynamic time warping (DTW) algorithm finds an optimal path between two time series. It compares each point of the first time series with a point in the second one. Hence, similar patterns occurring in different time slots are considered similar. A dynamic programming method finds the optimal match [99]. Here, we describe the DTW for k-dimensional time series. Algorithm 5.1 finds the distance between two k-dimensional time series with size of N and M.

5.3.2 Fuzzy hierarchical clustering

Given n time series $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, a fuzzy hierarchical clustering method finds a membership matrix $\mathbf{C} \in \mathbb{R}^{n \times c}$, where c is the number of clusters and $\mathbf{C}_{ij} \in [0, 1]$ is the fuzzy

Table 5.1: Multi-dimensional Dynamic Time Warping

1: **procedure** $\delta(\mathsf{a}, \mathsf{b})$ ▷ Distance of k-dimensional time series
2: **return** $\sum_{\mathsf{k}=1}^{\mathsf{K}} |\mathsf{a}[\mathsf{k}] - \mathsf{b}[\mathsf{k}]|$
3: **procedure** DTW($\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_\mathsf{N}\}, \mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_\mathsf{M}\}$) ▷ Two input time series
4: $\mathbf{X}, \mathbf{Y} \leftarrow \mathsf{Normalize}(\mathbf{X}, \mathbf{Y})$
5: $\mathbf{C}[1,1] \leftarrow \delta(\mathbf{x}_1, \mathbf{y}_1)$ ▷ Initialization of distance matrix
6: **for** $\mathsf{i} \leftarrow 2$ to N **do**
7: $\mathbf{C}[\mathsf{i}, 1] \leftarrow \mathbf{C}[\mathsf{i}-1, 1] + \delta(\mathbf{x}_\mathsf{i}, \mathbf{y}_1)$
8: **for** $\mathsf{j} \leftarrow 2$ to M **do**
9: $\mathbf{C}[1, \mathsf{j}] = \mathbf{C}[1, \mathsf{j}-1] + \delta(\mathbf{x}_1, \mathbf{y}_\mathsf{j})$
10: **for** $\mathsf{i} \leftarrow 2$ to N **do**
11: **for** $\mathsf{j} \leftarrow 2$ to M **do**
12: $\mathbf{C}[\mathsf{i}, \mathsf{j}] \leftarrow \mathsf{min}(\mathbf{C}[\mathsf{i}-1, \mathsf{j}], \mathbf{C}[\mathsf{i}, \mathsf{j}-1], \mathbf{C}[\mathsf{i}-1, \mathsf{j}-1]) + \delta(\mathbf{x}_\mathsf{i}, \mathbf{y}_\mathsf{j})$
13: **return** $\mathsf{d} \leftarrow \mathbf{C}[\mathsf{N}, \mathsf{M}]$ ▷ Return the distance of two time series

membership of a time series i to a cluster j. Here, a k-dimensional time series of a sensor i is represented with $\mathbf{x}_\mathsf{i}$. Also, distance of two time series is obtained by Algorithm 5.1.

To apply a DTW-based clustering method, the main challenge is to compute the mean of a cluster addressed in [49], [95], [99], because the initial values impact the output of the clustering method. Hence, we consider a fuzzy hierarchical clustering method without a need to find cluster means. We apply a fuzzy clustering method, with some modification compared with [65], on time series data. A complete-linkage distance function is used between two clusters, and a single-linkage is used between two time series, and a time series and a cluster.

An Algorithm 5.2 finds the membership matrix of time series to clusters. The matrix $\mathbf{D}$ is the set of distances between all pair of time series and clusters, and it is initialized by all distances between time series. The function $\mathsf{minDistance}(.)$ finds the closest pair of elements (a, b) in the set $\mathbf{D}$, which a and b can be a time series or a cluster. The matrix $\mathbf{C}$ is the list of clusters and their members. The function $\mathsf{updateClusters}(.,.,.)$ adds the selected pair of elements (a, b) to the list of clusters. This function merges a and b into a new cluster. The matrix $\mathbf{A}$ is the list of assigned time series, when a time series is assigned into one

cluster. Based on a new formation of clusters, the function $\mathsf{updateDistances}(.,.,.)$ finds the new distances between time series and clusters. It updates the distance of all clusters and unassigned time series to the new cluster. Moreover, it updates the fuzzy distance of all assigned time series to the new cluster, and all elements of the new cluster to other clusters. The fuzzy distance between assigned time series u, and a cluster $\mathbf{c_i}$ is obtained by using equations (5.2),

$$\begin{aligned} \mathsf{d}_{\mathsf{u}}^{\min} &= \min\{\mathsf{d}(\mathsf{u},\mathbf{c_j})|\mathbf{c_j}\in\mathbf{C}\} \\ \mu(\mathsf{u},\mathbf{c_i}) &= \frac{\mathsf{d}_{\mathsf{u}}^{\min}}{\mathsf{d}(\mathsf{u},\mathbf{c_i})+\mathsf{d}_{\mathsf{u}}^{\min}}, \\ \mathsf{d}(\mathsf{u},\mathbf{c_i}) &= \min\{(1-\log_{\mathsf{m}}(\mu(\mathsf{u},\mathbf{c_i})))*\mathsf{d}(\mathsf{u},\mathbf{c_i}),\mathsf{d}(\mathsf{u},\mathbf{c_i})\} \end{aligned} \tag{5.2}$$

where $\mathsf{d}_{\mathsf{u}}^{\min}$ is the minimum distance of a assigned time series u to any of the clusters, $\mu(\mathsf{a},\mathsf{b})$ is membership value of assigned time series u to the cluster $\mathbf{c_i}$, m is a fuzziness parameter, and the distance function $\mathsf{d}(.,.)$ is based on single-linkage method for each pair of time series, or a time series and a cluster, and complete-linkage method for two clusters.

Table 5.2: A DTW-based fuzzy hierarchical clustering on spatio-temporal data

procedure FHC($\mathbf{X}=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n\}$) ▷ n input time series
  $\mathbf{D}=\{\}$; ▷ Distance matrix
  $\mathbf{A}=\{\}$; ▷ List of assigned points
  $\mathbf{C}=\{\}$; ▷ List of clusters and their members
  $\mathbf{D}\leftarrow\mathsf{initializeDistances}(\mathbf{X})$;
  while Convergence is satisfied **do**
    $(\mathsf{a},\mathsf{b})\leftarrow\mathsf{minDistance}(\mathbf{D})$
    $\mathbf{C},\mathbf{A}\leftarrow\mathsf{updateClusters}(\mathbf{C},\mathbf{A},\mathsf{a},\mathsf{b})$
    $\mathbf{D}\leftarrow\mathsf{updateDistances}(\mathbf{D},\mathbf{A},\mathbf{C},\mathsf{a},\mathsf{b})$
  return C ▷ Return the list of fuzzy clusters

5.3.3 Convolutional layer

Convolutional layers use three primary ideas – local receptive fields, shared weights and spatial subsampling; making them effective and efficient models for extracting features from local stationary grid data [66]. Given an input matrix $\mathbf{X} \in \mathbb{R}^{n \times \bar{t} \times k}$, a 2-dimensional convolutional layer has a weight matrix $\mathbf{W} \in \mathbb{R}^{a \times b \times k}$, called a kernel, where $a \leq n$ and $b \leq \bar{t}$, and k is the number of channels. A convolutional multiplication $\mathbf{X} * \mathbf{W}$ is obtained by sliding a kernel over the input matrix. Strides s_1 and s_2 are used as the number of elements shifts over input on each dimension, e.g. it moves the kernel to s_1 elements at a time on the first dimension. The kernel is a shared weight in the convolutional layer. Given $\mathbf{X}^l$ as the input of layer l, the output is obtained by $\mathbf{Z}^l = \sigma(\mathbf{X}^l * \mathbf{W}^l + \mathbf{b}^l)$ for an activation function $\sigma(.)$ and a bias vector $\mathbf{b}^l$. Pooling layers are among successive convolutional layers, described by $\mathbf{X}^{l+1} = \mathsf{maxPool}(\mathbf{Z}^l)$. They reduce the size of hidden layers, while extracting features in locally connected layers. A max pooling layer with pooling size of (m, n) selects the maximum value in a matrix of size $\bar{\mathbf{W}} \in \mathbb{R}^{m \times n}$, and reduces the dimension of layers divided by m and n.

5.3.4 Convolution-LSTM layer

A Long-Short Term Memory (LSTM) is introduced to improve recurrent cells, when there is a long-term dependency in data [109]. A memory cell $\mathbf{c}^t$, input gate $\mathbf{i}^t$, output gate $\mathbf{o}^t$ and forgot gate $\mathbf{f}^t$ solve the exploding/vanishing gradient problem in recurrent layers. Given convolution operator $*$ and a Hadamard product $\circ$, a convolutional LSTM layer is formulated as follows [137],

$$
\begin{aligned}
\mathbf{i}^t &= \sigma(\mathbf{W}^{xi} * \mathbf{x}^t + \mathbf{W}^{hl} * \mathbf{h}^{t-1} + \mathbf{W}^{ci} \circ \mathbf{c}^{t-1} + \mathbf{b}^i) \\
\mathbf{f}^t &= \sigma(\mathbf{W}^{xf} * \mathbf{x}^t + \mathbf{W}^{hf} * \mathbf{h}^{t-1} + \mathbf{W}^{cf} \circ \mathbf{c}^{t-1} + \mathbf{b}^f) \\
\mathbf{c}^t &= \mathbf{f}^t \circ \mathbf{c}^{t-1} + \mathbf{i}^t \circ \tanh(\mathbf{W}^{xc} * \mathbf{x}^t + \mathbf{W}^{hc} * \mathbf{h}^{t-1} + \mathbf{b}^c) \\
\mathbf{o}^t &= \sigma(\mathbf{W}^{xo} * \mathbf{x}^t + \mathbf{W}^{ho} * \mathbf{h}^{t-1} + \mathbf{W}^{co} \circ \mathbf{c}^t + \mathbf{b}^o) \\
\mathbf{h}^t &= \mathbf{o}^t \circ \tanh(\mathbf{c}^t)
\end{aligned}
\tag{5.3}
$$

In other words, a convolutional-LSTM layer have same structure of convolutional layers, but with LSTM cells. The convolutional-LSTM layer has an input matrix $\mathbf{X} \in \mathbb{R}^{w \times a \times b \times k}$, where w is size of time window, and the matrix $\mathbf{W} \in \mathbb{R}^{a \times b \times k}$ is the spatial information on a grid of size a and b. Each element of the matrix $\mathbf{W}_{ij}$ has k features, represented as the number of channels.

5.3.5 A denoising stacked autoencoder

Given an input data $\mathbf{x} \in \mathbb{R}^m$, an autoencoder layer transforms input data with a non-linear function $\mathbf{z} = \sigma(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1), \mathbf{z} \in \mathbb{R}^d$, where $d < m$ is the latent representation size [126]. The decoder generates $\bar{\mathbf{x}} = \sigma(\mathbf{z}\mathbf{W}^2 + \mathbf{b}^2)$, where $\bar{\mathbf{x}} \in \mathbb{R}^m$ is the reconstruction of input data. In other words, the reconstruction of input data is obtained from latent feature $\mathbf{z}$, which represents the most important patterns. A dropout function is represented with $\mathbf{z}^{l+1} = \mathsf{dropout}(\ \sigma(\ \mathbf{z}^l\mathbf{W}^l + \mathbf{b}^l), \alpha)$, in which randomly α percent of neurons are dropped in each round of training step. In the training process, the objective is to reconstruct $\mathbf{x}$, by minimizing a loss function $L(.,.)$, e.g. least square function, between $\mathbf{x}$ and $\bar{\mathbf{x}}$ and obtaining optimum model parameters $\boldsymbol{\theta}^\star$ for all n input data points as follows,

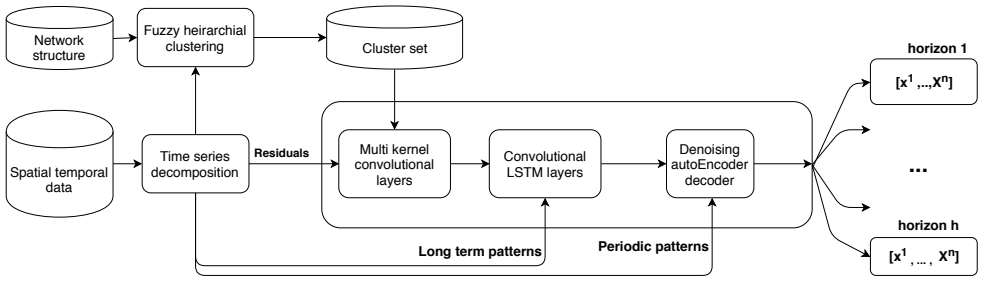


Figure 5.1: The proposed framework for the spatio-temporal forecasting problem

$$\boldsymbol{\theta}^{\star} = \text{argmin}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \mathsf{L}_{\boldsymbol{\theta}}(\mathbf{x}_i, \bar{\mathbf{x}}_i) \tag{5.4}$$

Stacked autoencoders are a set of multiple autoencoder layers, in which the input of each layer is the output of previous layer [127]. To have a robust model to missing values, the input data is corrupted with some missing values, while the output remains unchanged. Adding missing values to the input data and training the neural network to reconstruct the input data helps the neural network to be robust to missing values. Unsupervised training of stacked autoencoders can be used to reconstruct the original data in the presence of noisy or missing data [150], [46], [13]. In this chapter, we use a pretrained autoencoder to improve the performance of the neural network in existence of missing data.

5.4 Methodology

In this section, we describe the architecture of the proposed deep learning framework, in Fig. (5.1).

5.4.1 Preprocessing

In our proposed framework, the preprocessing of data includes several steps, represented in Fig. 5.1. A time series decomposition method is applied on input time series $\mathbf{X} \in \mathbb{R}^{n \times w \times k}$. It generates three time series components of $\mathbf{X} = (\mathbf{S}, \mathbf{T}, \mathbf{R})$, which are seasonal, trends and residuals of time series data, respectively. In spatio-temporal data, time series residuals not only represent random noise, but also can capture spatial patterns. For example, in a transportation network, time series residuals can be caused by the traffic evolution of the transportation network and they are meaningful patterns in a spatial neighborhood. We illustrate the existence of spatial patterns in time series residuals, in Section 5.5.

To apply Algorithm 5.2 on time series residuals, we consider a set G for geographically nearest neighbors of sensors. As the objective is to have smooth clusters over a geographical area, the algorithm updates single-linkage distances between two time series from set G; thus one cluster would not be distributed in a geographical area. The output of the clustering algorithm is a fuzzy membership of each sensor to their clusters. Each sensor $\mathbf{x}_i$ has a membership to multiple clusters. A DTW distance function finds the non-linear similarity between any two time series $\mathbf{x}_i$ and $\mathbf{x}_j$. A fuzzy hierarchical clustering algorithm finds the cluster of sensors with similar time series residuals by finding the clusters in which the distance of its members is minimized. To represent short-term similarity among neighboring time series, we use a sliding window method on training data and obtain an average of the corresponding DTW distances. A sliding window method finds similarities between short-term time windows of neighboring areas. To reduce computational time, the sliding window method is only applied when there is high similarity among neighboring time series. For example, in traffic flow data, the interaction among neighboring sensors increases in peak hours and congested time periods. Applying Algorithm 5.2 with the aforementioned modifications on spatial time series finds fuzzy clusters of time series, stored in the cluster set in Fig. 5.1.

5.4.2 Neural network models

The architecture of the deep neural network is presented in Fig. 5.2. The spatio-temporal data is a high dimensional time series data, in which there is a spatial correlation among neighboring time series. Hence, the model has two main components. The convolutional component captures spatial patterns, and the recurrent component captures temporal patterns. Time series residuals are the first input of the neural network, detrended and represented by a matrix $\mathbf{R} \in \mathbb{R}^{s \times w \times k}$. A convolutional component is applied to extract patterns from the time series residuals. The spatial relation among time series can be a grid structure, e.g. weather data, or a graph structure, e.g. traffic data. In a general convolutional layer, a kernel slides on the first and second dimensions. However, because the sensors can have a graph structure relation, like sensors in a transportation network, sliding a kernel on spatial features, the first dimension, looses the structure of the network [74]. Hence, we propose a multi-kernel convolutional layer, which receives the cluster set and time series residuals. A kernel $\mathbf{W}_i$ is assigned to a cluster i, and it is described with weight matrix $\mathbf{W}_i$ where $\mathbf{W}_{ij} \neq 0$, if $j \in c_i$. In other words, the size of trainable variables for a kernel, corresponding to cluster i, is $\mathbf{W}_i \in \mathbb{R}^{|c_i|,w,k}$. Only the sensors in cluster i, have local connectivity to corresponding time series residuals. Each kernel only slides over the temporal axis and obtains hidden units $\mathbf{h}_i = \mathsf{maxPool}(\, \sigma(\, \mathbf{R}_i \mathbf{W}_i + \mathbf{b}_i))$ for all $i \in \{1, \ldots, |\mathbf{C}|\}$. Several convolution-Rectified Linear Unit(ReLU)-Pooling layers extract short-term and spatial patterns from the time series residuals in each neighborhood. The output of the kernels are concatenated and connected to a fully-connected layer $\mathbf{h}^{l+1} = \mathsf{concat}(\mathsf{FC}(h_1^l), \ldots, \mathsf{FC}(h_{|\mathbf{C}|}^l))$ and represented with a hidden layer $\mathbf{h}^{l+1} \in \mathbb{R}^{w \times s \times v \times 1}$, where v is the number of represented features in convolutional layers and s is the total number of sensors.

The time series trends represent long-term patterns, the second input of the proposed neural network. The trends of time series $\mathbf{T}$ concatenate to $\mathbf{h}^{l+1}$ on the last axis, $\mathbf{h}^{l+2} = \mathsf{concat}(\mathbf{h}^{l+1}, \mathsf{FC}(\mathbf{T}), \mathsf{axis} = 4)$ which results in $\mathbf{h}^{l+2} \in \mathbb{R}^{w \times s \times v \times 2}$. Time series residuals are only

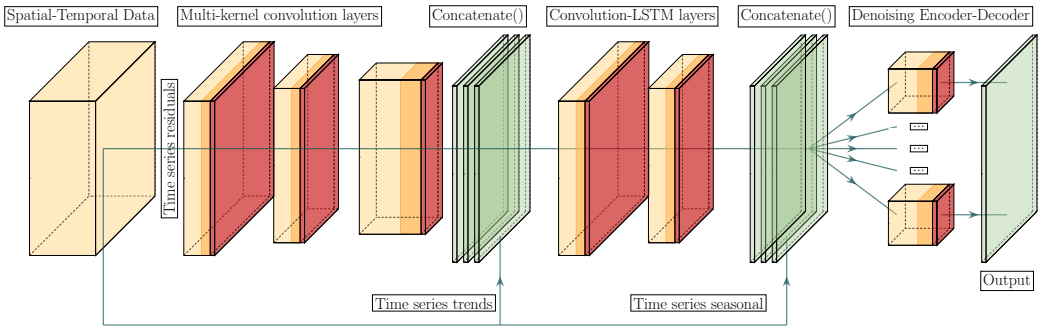


Figure 5.2: The proposed spatial-temporal decomposition deep neural network architecture

similar in a small spatial neighborhood. However, time series trends can represent traffic patterns in a larger geographical area, examined in Section 5.5.2. Hence, we consider Conv-LSTM layers to capture spatio-temporal patterns of the output of multi-kernal convolutional layer, concatenated with time series trends. This layer, described in section 5.3.4, receives an input $\mathbf{h}^{l+2}$, and apply the kernel on the matrix of size $(\mathsf{a} = \mathsf{s}, \mathsf{b} \leq \mathsf{v})$ with two channels. This convolutional layer has different architecture with the first multi-kernel convolutional layer, that is, each neural cell is an LSTM cell and is applied on all input sensors. The convolutional LSTM layers extract features from residuals and trends.

Spatio-temporal data can include seasonal patterns, e.g. traffic flow data have weekly seasonal pattern, represented in Section 5. The output is concatenated with seasonal patterns of time window of size $\mathsf{w} + \mathsf{h}$, represented with $\{\mathsf{t} - \mathsf{w}, \ldots, \mathsf{t} + \mathsf{h}\}$. It is followed by a fully-connected layer. The output is $\bar{\mathbf{y}} \in \mathbb{R}^{\mathsf{s} \times \mathsf{h} \times \mathsf{k}}$, where h is the prediction horizon. The output $\bar{\mathbf{y}}$ consists of predicted values for all sensors in the prediction horizon.

One of the challenges in spatio-temporal data is to have robust prediction with missing or noisy data. In real-time data obtained by sensors over a geographical area, missing data or noisy data are highly probable. We consider a separate component at top of the proposed model for robust prediction of missing data. An encoder decoder component is the last component of the model. A denoising encoder decoder reconstructs the last output $\bar{\mathbf{y}}$ for each cluster. In the pretraining step, for a prediction horizon h and a cluster j, each autoencoder

generates $\bar{\mathbf{x}} = \mathsf{DA}_{\mathsf{j}}(\mathbf{x})$, where $\bar{\mathbf{x}} \in \mathbb{R}^{\mathsf{s}\times\mathsf{h}\times\mathsf{k}}$ is the reconstruction of input. An autoencoder component predict traffic flow $\bar{\mathbf{y}}_{\mathsf{d}} = \mathsf{DA}(\bar{\mathbf{y}})$. As the output of autoencoders is designed based on the clusters, there are some sensors $\mathbf{x}_{\mathsf{k}} \in \mathbf{c}_{\mathsf{i}} \cap \mathbf{c}_{\mathsf{j}}, \mathsf{i} \neq \mathsf{j}$, where the fully-connected target layer FC^{t} is connected to all common variables between denoising autoencoders with a linear activation function $\mathbf{y}_{\mathsf{output}} = \mathsf{FC}^{\mathsf{t}}(\mathsf{DA}_1(\bar{\mathbf{y}}), \ldots, \mathsf{DA}_{|\mathsf{C}|}(\bar{\mathbf{y}}))$. In other words, the final layer finds the weighted average of predicted values of a sensor, if it belongs to more than one cluster. In the training process the objective is to minimize the loss function $\mathsf{L}(.,.)$, such as a mean square error function, between $\mathbf{y}_{\mathsf{output}}$ and actual values $\mathbf{y}$, and to obtain optimum model parameters $\boldsymbol{\theta}^{\star}$ for input data using stochastic gradient descent,

$$\boldsymbol{\theta}^{\star} = \text{argmin}_{\boldsymbol{\theta}} \sum_{\mathsf{i}=1}^{|\mathbf{X}|} \mathsf{L}_{\boldsymbol{\theta}}(\mathbf{x}_{\mathsf{i}}, \mathbf{y}_{\mathsf{i}}) \tag{5.5}$$

5.5 Experimental analysis

In this section, we apply the proposed model on traffic flow data. We analyze spatio-temporal patterns, and evaluate the performance of the proposed model compared with baseline and state-of-the-art neural network models.

5.5.1 Dataset

We use traffic flow data from the Bay Area of California represented in Fig. 5.3 which have been broadly used, and available in PeMS [1]. The traffic data is gathered every 30 seconds and aggregated over 5 minute periods. Each sensor on highways has flow, occupancy and speed at each time stamp. A sensor is an inductive loop traffic detector device on mainline, off-ramp or on-ramps locations. In a preprocessing step, we select 597 sensors which have

more than 90% of the observed values in the first six months of 2016. The neural network models receive all sensors' data as input, and predict the value of mainline sensors, 380 of sensors, for a given prediction horizon. For this analysis, a large number of sensors are selected to better evaluate the proposed model. A deep learning model generally outperforms statistical and traditional machine learning models, when there is a large-scale training data with complex patterns [143]. Moreover, such a large number of sensors can illustrate the generalization of the proposed model over various types of spatial and temporal patterns.

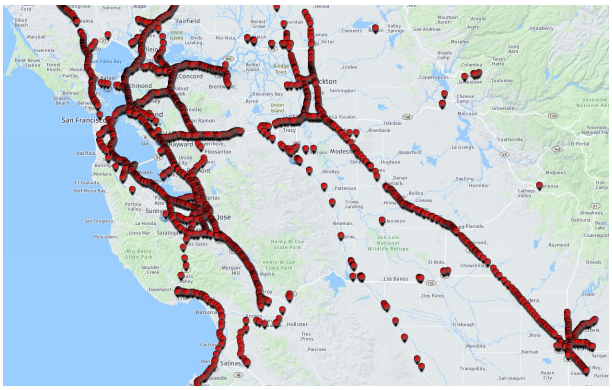
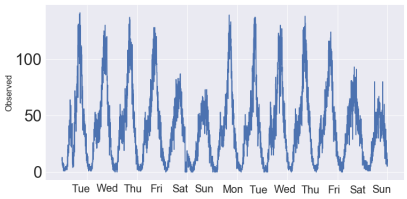


Figure 5.3: The red dots represent loop detector sensors on highways of Bay Area, California.

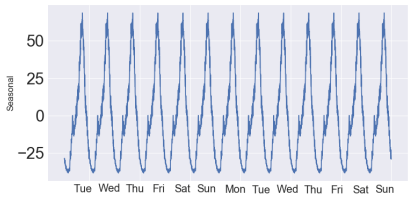
5.5.2 Pattern analysis in traffic data

Spatial patterns in traffic data are the results of traffic evolution in the network. Here, we analyze the spatial, short-term and long-term patterns. In Fig. 5.4, an additive time series decomposition of traffic flow data is presented for one station. Given a one day frequency, time series decomposition has similar, repeated (seasonal) daily patterns. Moreover, there are long-term weekly patterns, shown as trends $\mathbf{T}$. The long-term patterns, such as seasonal and trends, arise from similar periodic patterns, generated outside of the highway network. In other words, they are related to origin-destination matrix of vehicles in the network. The time series residuals are not only random noise, but also the results of spatial and short-term patterns, related to the evolution of traffic flow or sudden changes in smaller regions of the network. Hence, time series residuals of neighboring sensors are more similar with each

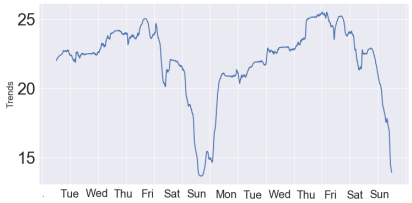
other.



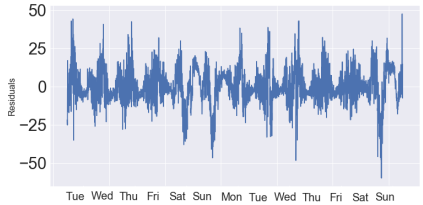
(a) The observed traffic flow data.



(b) Seasonality of traffic flow data



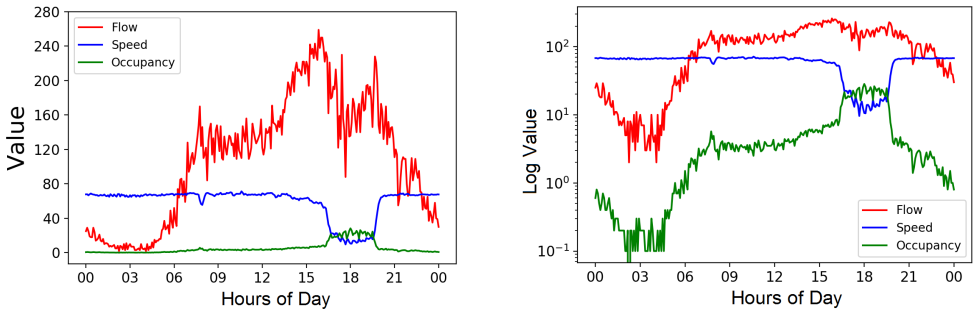
(c) Trends of traffic flow data



(d) Residuals of traffic flow data

Figure 5.4: Time series decomposition with daily frequency is represented for one sensor's traffic flow data.

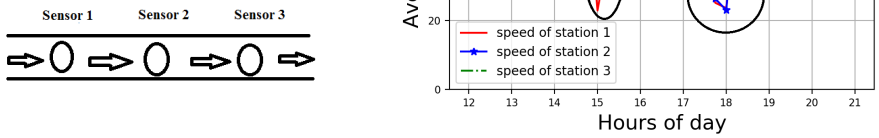
Time series residuals are interpreted as random noise for time series data. However, in traffic flow data, the residuals are the results of traffic evolution in the network. In Fig. 5.5, we examine the non-linear relation of flow, speed and occupancy in one day and one sensor. It shows that high occupancy reduces speed in a road segment, which is the result of traffic congestion. For more details, we refer the reader to the theoretical analysis of these relationships in [38] and [89], respectively. In a transportation network, the congestion propagation describes the relation among neighboring sensors of a highway, shown in Fig. 5.6. For a given three sensors, traffic congestion is propagated with nearly 20 minutes of delay. For a larger geographical area, the speed of 13 successive sensors is represented in an image-like representation of spatio-temporal data, in Fig. 5.7. The reduction of speed in peak hours is presented with darker colors. It shows that the reduction in speed is similar in neighboring areas, which also represents the existence of spatial correlation in neighboring sensors.



(a) The relation between flow, occupancy and speed is shown. Occupancy, with value more than 8% occupied by vehicles, decreases average speed, which is the result of traffic congestion.

(b) Log plot to represent the linear relation between occupancy and flow in free flow speed (about 70 mph).

Figure 5.5: The relation between occupancy, speed and flow



(a) Three successive sensors are selected to represent congestion propagation in the network.

(b) The reduction in speed of sensor 1 and 2 can be observed twice in this plot, in which there is 20 minute delay due to congestion propagation delay time.

Figure 5.6: The congestion propagation in successive sensors.

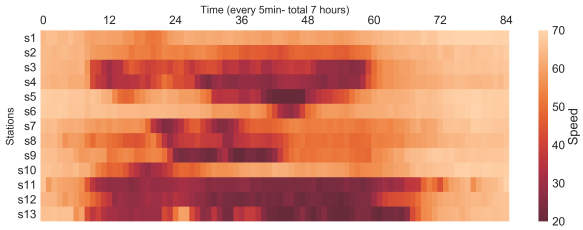


Figure 5.7: Image-like representation of a speed value of 13 successive sensors over 7 hours (5 min time stamp)

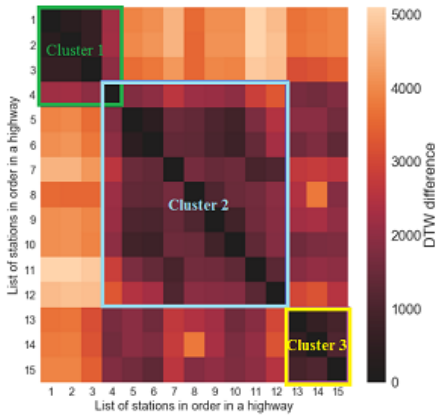


Figure 5.8: The table shows the Dynamic Time Warping distance of time series residuals among 15 sensors on a highway. The result of hierarchical clustering method is illustrated with three clusters. The distance values near to diagonal have lower values, as they are more similar with each other.

5.5.3 Fuzzy hierarchical clustering

In this section, we illustrate the results of a fuzzy clustering method, applied on the time series residuals. In Fig. 5.8, the DTW distance matrix shows the similarity among time series residuals of neighboring sensors. The matrix shows that the average DTW distance for peak hours has the highest dependency among neighboring sensors. Each fuzzy cluster is obtained by applying Algorithm 5.2 on the training data. On the elements near the diagonal, the lowest distance values show the similarity between neighboring sensors. The fuzzy clustering method finds the membership of each sensor to the clusters. In the fuzzy membership matrix, we consider a threshold of 0.1. All the sensors with a membership value of more than 0.1 are considered as the members of the clusters. We also consider the average length of clusters to be less than 10 miles. The agglomerative clustering method stops when the average size of clusters become greater than 10. As the clustering method is only applied to mainline sensors, we also add the on-ramp and off-ramp sensors to the cluster of closest mainline sensor. The output of fuzzy hierarchical clustering method has 64 clusters, where the average number of elements is 9.7 with standard deviation of 4.2 and minimum cluster size of 3 and maximum of 14. The length of smallest and largest cluster is 0.3 mile and 32.1

mile. And there are 53 sensors which appear in more than one cluster, nearly 10% of total sensors.

To examine the relation between trends in one spatial area, we obtain DTW distance of each pair of sensors. A sliding window method generates time series sequences. For a given time window, we normalized trends by subtracting all time stamp values from the last value. The average DTW distance is 0.7, for all pairs of sensors, which shows the high similarity of trends. By contrast, the average DTW distance of time series residuals is 4.5 for all pair of sensors, while applying the fuzzy clustering method on time series reduces the average DTW of clusters to 0.6. This analysis shows that similarity among trends is independent of distance of sensors, while similarity of time series residuals significantly increases for neighboring sensors. Hence, we only apply fuzzy clustering on time series residuals.

5.5.4 Comparison of results

Here, we describe other implemented models, compared with the proposed deep learning model. All of the neural network models are trained using the ADAM optimizer. The batch size and epochs are set to 512 and 400, respectively. All experiments are implemented with Keras [33]. We used a grid search method for finding the optimum deep neural network architectures which have the best performance and most efficient computational time.

The input matrix is $\mathbf{X} \in \mathbb{R}^{\mathsf{s} \times \mathsf{w} \times \mathsf{k}}$, where the number of sensors is $\mathsf{s} = 597$, the time window is $\mathsf{w} = 6$, and there are $\mathsf{k} = 3$ features, including flow, occupancy and speed. For FCNN, LSTM, CNN and the proposed multi-kernel CNN-LSTM models, the input dimension is reshaped to have appropriate dimensions. The data for all models is scaled into range of [0, 1]. For the models without a time series decomposition component, including FCNN, LSTM and CNN, we transform the non-stationary data into stationary data by subtracting all input values, from time stamp $\mathsf{t} - \mathsf{w}$ to time stamp t, and output values, from time stamp $\mathsf{t} + 1$ to $\mathsf{t} + \mathsf{h}$,

from the value at time step t. Detrending of time series for the models with decomposition component is as follows. The residual time series are stationary. To feed trends and seasonal components to a neural network, we make them stationary by subtracting each time window from its last value $\mathbf{S}^{\mathsf{t}}$ and $\mathbf{T}^{\mathsf{t}}$. To recover the output, we add the predicted value to sum of $\mathbf{S}^{\mathsf{t}}$ and $\mathbf{T}^{\mathsf{t}}$. The output matrix is $\mathbf{Y} \in \mathbb{R}^{\mathsf{s}\times\mathsf{h}\times\bar{\mathsf{k}}}$, where the size of horizon $\mathsf{h} = 4$ for 15-min, 30-min, 45-min and 60-min prediction in the resulting tables and $\bar{\mathsf{k}} = 1$ only for traffic flow prediction.

As illustrated in primary traffic flow prediction studies, the traffic flow patterns are similar in the same hours and weekdays. The first baseline model (Ave-weekday-hourly) is to use average of traffic flow of each sensor as a time-table for each time t, given a weekday and hours of day. The short-term prediction for each sensor is obtained using average values in the training data. The second baseline model (current value) is to use current value of traffic flow $\mathbf{x}^{\mathsf{t}}$ for the prediction horizon $\mathbf{x}^{\mathsf{t}+\mathsf{h}}$.

In this section, we describe the implemented neural network models. A fully-connected neural network (FCNN) architecture with three fully connected layers of (500, 300, 200) hidden units, Xavier initialization, and an ReLU activation function is implemented. A DBN with greedy layer wise pretraining of autoencoders finds a good initialization for a fully-connected neural network. A fine tuning step for stacked auto encoder finishes the training. We consider 30 epochs for pretraining each layer. An LSTM neural network is capable of capturing long-term temporal patterns. However, in most of the studies, LSTM models have a shallow architecture, because they are slow to converge. An LSTM model with two layers of 400 and 200 hidden units is implemented. The hidden unit of second layer is connected to a fully connected layer. Also the input is reshaped from a vector to a matrix of two dimensions $(\mathsf{w}, \mathsf{s} \times \mathsf{k})$. To use a convolutional neural network, the input matrix is reshaped to three dimensions $(\mathsf{w}, \mathsf{s}, \mathsf{k})$. The first two dimensions are image-like representation of time series data and the third dimension is the channel, which includes

traffic flow, speed and occupancy. The optimum implemented deep CNN model has four layers with a max-pooling and a batch normalization. The number of filters are (16, 32, 64, 128), the kernel size is (5, 5), and max-pooling layers reduce the dimension by two in each layer. It follows by two fully connected layers.

The CNN-LSTM model captures short-term and spatial patterns in convolutional layers, and temporal patterns in LSTM layers. Two convolutional layers are applied on all input sensors with filters of size (16, 32). The output of CNN layer is connected to LSTM layer. An LSTM layer has the size of (300, 150) units, followed by a fully connected layer. The model C-CNN-LSTM is a clustering based CNN-LSTM, in which a multi-kernel convolutional layer extracts spatial, short-term patterns from time series residuals. The clusters are obtained in Section 5.5.3.

A pretrained autoencoder decoder is applied to each cluster of sensors to generate a robust output. Each layer is connected to a dropout layer with rate of 0.3. As the average size of clusters is nearly 10 and standard deviation is 4, described in section 5.5.3, we use a same architecture for all of the clusters, which has 5 fully-connected layers and the size of (40, 20, 10, 20, 40) units, and an ReLU activation function. The pretraining has 60 epochs. The weights are loaded into the proposed model in fine tuning step.

The cluster based CNN-LSTM with an autoencoder (C-CNN-LSTM-DA) is the proposed model in section 5.4 which uses clustering of time series residuals, trends, and seasonal along with an autoencoder. The proposed architecture, in section 5.4, consists of two convolutional layers with ReLU and max-pooling layers with filters of size (32, 64). It follows by two fully connected layers, two 2-dimension convolutional LSTM for capturing long-term patterns with hidden units of size (16, 32).

5.5.5 Performance metrics

To evaluate the performance of the proposed model, mean absolute error (MAE), mean absolute percentage error (MAPE) and root mean square error (RMSE) are used as performance metrics, defined in equations 5.6,

$$\begin{aligned}
\mathsf{MAE}(\mathbf{y}, \bar{\mathbf{y}}) &= \frac{1}{\mathsf{n}} \sum_{\mathsf{i}=1}^{\mathsf{n}} |\mathsf{y_i} - \bar{\mathsf{y}}_\mathsf{i}| \\
\mathsf{MAPE}(\mathbf{y}, \bar{\mathbf{y}}) &= \frac{1}{\mathsf{n}} \sum_{\mathsf{i}=1}^{\mathsf{n}} \frac{|\mathsf{y_i} - \bar{\mathsf{y}}_\mathsf{i}|}{\mathsf{y_i}} \times 100 \\
\mathsf{RMSE}(\mathbf{y}, \bar{\mathbf{y}}) &= \sqrt{\frac{1}{\mathsf{n}} \sum_{\mathsf{i}=1}^{\mathsf{n}} (\mathsf{y_i} - \bar{\mathsf{y}}_\mathsf{i})^2}
\end{aligned} \tag{5.6}$$

where actual values are $\mathbf{y}$ and predicted values are $\bar{\mathbf{y}}$. Also, the number of data points in testing data is $\mathsf{n} = \bar{\mathsf{t}} \times \mathsf{s}$, where $\bar{\mathsf{t}}$ is the number of time stamps or temporal features, and s is the number of sensors or spatial features. These performance metrics are a measure of similarity between actual traffic flow and predicted traffic flow. In spatio-temporal data, the prediction error can dramatically change over various spatial or temporal features, e.g. some of the locations are more difficult to predict, or the prediction error increases in peak hours with high traffic fluctuations. Hence, we define statistical indicators to evaluate the performance of the models over spatial and temporal features.

5.5.6 Spatial performance metrics

To evaluate the performance of models over spatial features, we describe a spatial statistical indicator. A spatial standard deviation is obtained by $\mathsf{sstd} = \sqrt{\frac{1}{|\mathbf{S}|} \sum_{\mathsf{i} \in \mathbf{S}} (\mathsf{e_i} - \bar{\mathsf{e}})^2}$, where $\mathbf{S}$ is the set of spatial features or sensors, $\bar{\mathsf{e}} = \frac{1}{|\mathbf{S}|} \sum_{\mathsf{i} \in \mathbf{S}} \mathsf{e_i}$ is the mean error, and $\mathsf{e_i}$ is prediction

Table 5.3: Evaluation of the models for the traffic flow forecasting problem.

		Baseline models		neural networks models				proposed models		
Horizon	Metric	current-value	Ave-weekday-hourly	FCNN	DBN	LSTM	CNN	CNN-LSTM	C-CNN-LSTM	C-CNN-LSTM-DA
15 min	MAE	24	27.1	16.3	15.5	14	16	13.6	12.3	**12.1**
	RMSE	36	43.2	28.1	27	25	27.4	24.8	23.1	**22.7**
30 min	MAE	31	27.1	16.9	15.9	14.4	16.2	14.3	12.7	**12.4**
	RMSE	45	43.2	29	28	26.2	28.4	26.0	23.4	**22.9**
45 min	MAE	38	27.1	17.1	16.2	14.9	16.8	15	12.9	**12.8**
	RMSE	54	43.2	29.8	29	28.1	29.3	28.2	23.4	**23.1**
60 min	MAE	44	27.1	17.6	16.5	15.2	17.2	15.1	**13.3**	**13.3**
	RMSE	63	43.2	30.8	29.3	28.4	30.1	28.1	23.8	**23.7**

error in sensor i. Also, the maximum and minimum of $\mathsf{e_i}$ is represented, as the best case and worst case prediction error over various locations. We also define $\mathsf{p}\% - \mathsf{error}$ as the number of sensors, out of 380 sensors, where their prediction error is at least p percent better than other models' prediction error. This analysis illustrate the performance of the proposed model over various spatial features in Section 5.5.9. It also shows that the model is well generalized over various locations of a transportation network.

5.5.7 Temporal performance metrics

In Section 5.5.10, statistical indicators are defined to represent the performance of the proposed model over temporal features. A temporal standard deviation is obtained by $\mathsf{tstd} = \sqrt{\frac{1}{|\mathbf{T}|}\sum_{\mathsf{t}\in\mathbf{T}}(\mathsf{e}^{\mathsf{t}} - \bar{\mathsf{e}})^2}$, where $\mathbf{T}$ is the set of time slots, $\bar{\mathsf{e}}$ is the mean error and e^{t} is the error at time stamp t. Because the performance of the forecasting models can vary in different traffic states, various types of time slots $\mathbf{T}$ are selected to evaluate TSTD. As our case study is traffic flow data, and there is a weekly periodic patterns, we consider $\mathbf{T}$ as week days and hours of a day. Also, a split on off-peak hours and peak hours is examined in Section 5.5.10.

Table 5.4: Spatial statistical indicators for 15-min traffic flow forecasting.

		Spatial statistical indicators				Number of sensors out of 380		
Models	Metrics	Mean	SSTD	Min	Max	1%-error	5%-error	10%-error
FCNN	MAE	16.3	8.7	8.0	33.2	0	0	0
	RMSE	28.1	14.1	10.8	53.0	0	0	0
	MAPE	16	3.0	11.2	20	0	0	0
CNN-LSTM	MAE	13.6	7.2	7.3	28.5	0	0	0
	RMSE	24.8	11.5	9.8	45.3	0	0	0
	MAPE	14.8	2.4	11.0	19	0	0	0
C-CNN-LSTM-DA	MAE	12.4	5.8	6.8	24.6	380	366	301
	RMSE	22.9	9.8	9.5	43.2	372	328	281
	MAPE	13.8	1.9	10.5	17.6	334	269	203

5.5.8 Performance results on test data

In the first analysis, we compare the performance of models for the traffic flow prediction problem. The results are illustrated in Table 5.3. The comparison is for prediction horizons of 15-min, 30-min, 45-min and 60-min. In this section, mean of prediction error is the performance metric. The training errors are very close to testing errors. It shows that there is a low bias and variance in the neural network models. Hence, to briefly illustrate the results, we only represent the testing error throughout experimental results. The first model, current-value, has a good performance for very short-term prediction horizon; however, its performance dramatically reduces for longer prediction horizons. The second baseline model uses the average of weekly-hourly table. It has the lowest performance on this dataset, although there are repeated weekly patterns on traffic flow data. The performance of neural network models are significantly better than the baseline models. The LSTM model has a better performance than FCNN, BN and CNN models, demonstrating its strength in time series forecasting problems. CNN-LSTM models better capture short-term and long-term patterns. Two models, C-CNN-LSTM and C-CNN-LSTM-DA, have better performance due to explicitly separating spatial regions. The performance of C-CNN-LSTM and C-CNN-LSTM-DA is almost quite close. In next sections, we illustrate that the model with the autoencoder component has a better performance in existence of missing data.

5.5.9 Performance results over spatial features

Prediction error of the neural network models can be different over spatial features, as they have different patterns and structures. To illustrate the performance of models over spatial features, in Table 5.4, the spatial statistical indicators of forecasting models are presented. To briefly show the results, we only consider FCNN, CNN-LSTM and C-CNN-LSTM-DA for 15-min prediction horizon. As illustrated in Section 5.5.6, mean, spatial standard deviation, minimum and maximum of prediction errors are presented. Also, for each model, the number of sensors which have $\mathsf{p}\%$ lower prediction error than other two models is illustrated. Spatial standard deviation error represents that the prediction error significantly is different in various locations. The reason is that traffic flow data have different characteristics in different locations. Also, the minimum and maximum prediction error of the proposed model is lower than the other models. The results show that 380 sensors, out of 380 sensors, have at least 1% lower MAE than other models. Also 366 and 301 sensors have 5% and 10% lower MAE than other models, respectively. Zero values of $\mathsf{p}\% - \mathsf{error}$ for FCNN and CNN-LSTM models show that there is not any sensor with at least 1% lower error than the proposed model. These analyses on MAE, RMSE and MAPE illustrate that the proposed forecasting model, not only have better mean prediction error, but also its performance is better over all spatial features.

5.5.10 Performance results over temporal features

Statistical indicators of temporal features are evaluated in this section. A temporal standard deviation error (TSTD) is defined in section 5.5.7. We evaluate temporal statistical indicators in different week days. The proposed model has MAE of 12.4, 12.8, 12.3, 12.2, 12.0, 11.6, 11.8 for 15-min prediction horizon over 7 weekdays. The standard deviation is 0.37. Also, the standard deviation of MAE for FC-NN and CNN-LSTM is 0.46 and 0.59. These low

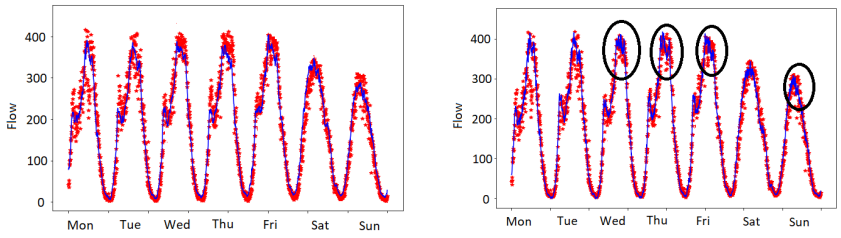
Table 5.5: Temporal statistical indicators for 15-min traffic flow forecasting.

		Temporal statistical indicators			
Models	Metrics	Mean	TSTD	Min	Max
FCNN	MAE	16.3	11.9	2.1	35.0
	RMSE	28.1	21.3	3.1	54.1
	MAPE	16	8.1	1.1	41.3
CNN-LSTM	MAE	13.6	11.0	2.1	31.7
	RMSE	24.8	20.5	3.0	52.4
	MAPE	14.8	6.4	1.0	36.0
C-CNN-LSTM-DA	MAE	12.4	9.8	1.9	29.1
	RMSE	22.9	17.3	3.0	45.1
	MAPE	13.8	4.9	1.0	29.1

values of temporal standard deviations represent that performance of these models is almost the same in different days of a week. However, in the rest, we illustrate that the prediction error has a high standard deviation over different hours of the day.

In Table 5.5, the statistical indicators over various weekly-hourly time stamps is examined, as traffic flow data have repeated patterns over one week. The available dataset has 5-min time stamps, 288 time stamp per day. The total number of weekly-hourly time stamps are 7 multiplied by 288. The average prediction error over such time stamps is obtained, and statistical indicators is illustrated in the table. The high value of temporal standard deviation error illustrate high variation of prediciton error over hours of a day. It shows that prediction error significantly changes in different traffic states. Minimum prediction error occurs during low traffic flow values and all models almost have same minimum values. The output of models is comparable in such traffic flow states. However, CNN-LSTM models outperform other models in peak hours, when there is a high fluctuation in the data and it has a more complex structure. The result of maximum prediction errors in three examined models illustrate that the proposed model significantly reduces the maximum prediction error over weekly-hourly time slots.

Next experiment is to compare peak and off-peak hours. In peak hours, the evolution of traffic flow possibly affect congestion propagation in network. Therefore, the time series residuals come from the evolution of traffic and are meaningful spatial patterns. On the other hand,



(a) Traffic flow prediction with FCNN(b) Traffic flow prediction with C-CNN-LSTM-DA

Figure 5.9: An example of traffic flow data for one sensor over one week is shown, where the blue line is predicted values, and the red dots are the actual values. The proposed model outperforms FCNN in peak hours, while they have comparable performance in off-peak hours. The black circles represent peak hours, where the predicted values are closer to actual values in the proposed model.

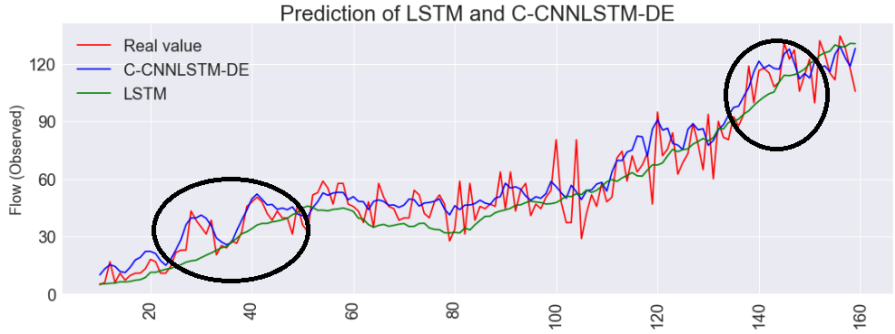


Figure 5.10: The proposed model can better capture residual patterns. Some of the big fluctuations are meaningful residual patterns, and can be predicted.

in off-peak hours, traffic flow is based on free flow speed and without any congestion. Such traffic state is easily predictable by most of the models. In Fig. 5.9, the output of the C-CNN-LSTM-DA and FCNN models are shown. Among neural network models, the FCNN model has the worst traffic flow prediction performance, while C-CNN-LSTM-DA is the best in Table 5.3. The C-CNN-LSTM-DA model has better performance in peak hours of the days, when there are high values of time series residuals. It shows the weakness of FCNN for capturing spatial patterns. In off-peak hours, all neural network models have comparable performance, as there is not complex patterns in the data.

In Table 5.6, we compare the performance of the models for peak and off-peak hours. We select FCNN, LSTM and the proposed model. This table compares residual-MAE and MAE.

For residual MAE, we detrend traffic flow prediction and find MAE error. For off-peak hours, LSTM has a comparable performance to the proposed model, as it simply capture long-term patterns. However, the performance of C-CNN-LSTM-DA is significantly better than the LSTM model in peak hours. In Fig. 5.10, we plot the comparison of LSTM and C-CNN-LSTM-DA. It shows that the C-CNN-LSTM-DA model captures big values of residuals compared to the LSTM model, as they are the result of traffic congestion, and not a random noise. The results show that the proposed model is better capable of capturing spatial patterns than the LSTM model.

Table 5.6: Performance evaluation of three models for traffic flow forecasting in peak and off-peak hours

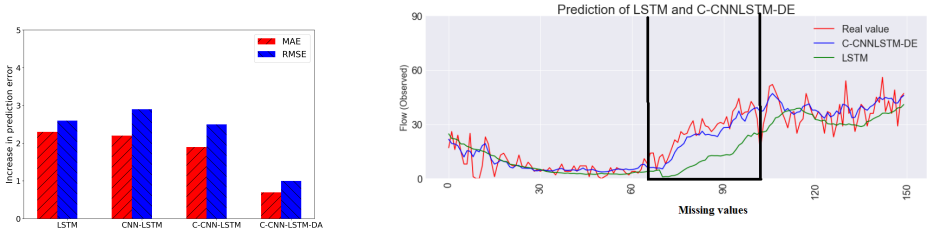
Flow State	Metric	MLP	LSTM	C-CNN-LSTM-DA
Off-peak hours	Residual MAE	6.1	4.3	4.4
	MAE	12.3	11.9	11.8
Peak hours	Residual MAE	12.2	11.1	8.2
	MAE	18.3	16.8	13.8

5.5.11 Performance results with missing data

Here, we evaluate the performance of the models, when there is missing data. In preprocessing steps, we select the sensors which do not have missing data. Then, we generate random blocks of missing values in the test data. Each block is related to one randomly selected sensor at a random starting time. The length of missing blocks are generated with a normal distribution with mean value of 2 hours and standard deviation of 0.5. The performance of the models is evaluated on missing values.

The results are illustrated in Table 5.7. To briefly describe the results, we only show the 30-min prediction horizon. The performance of the C-CNN-LSTM-DA model is better than other models. However, in Table 5.3, the performance of C-CNN-LSTM and C-CNN-LSTM-DA are very close. The results show that C-CNN-LSTM-DA is more robust to missing data. In Fig. 5.11.a, we illustrate that the prediction error of the LSTM model increases in

existence of missing data. The figure shows a reduced increase of prediction error in C-CNN-LSTM-DA. In other words, the plot shows the increase of the prediction error in existence of missing data. Fig. 5.11.b shows the difference of predicted values between LSTM and C-CNN-LSTM-DE.



(a) The increase in prediction error is illustrated.

(b) In a block of missing values, the prediction of the LSTM and the proposed model are illustrated.

Figure 5.11: Prediction results with random missing data

Table 5.7: The MAE and RMSE of forecasting models with randomly generated missing data.

Metric	LSTM	CNN-LSTM	C-CNN-LSTM	C-CNN-LSTM-DA
MAE	16.7	16.5	14.1	13.1
RMSE	28.8	28.9	25.2	23.9

5.5.12 Hypothesis testing

In the previous sections, the performance of deep learning models are evaluated using some standard performance metrics, such as MAE and RMSE. In addition, we define temporal and spatial statistical indicators, and evaluate forecasting models over different spatio-temporal states. The results show that the proposed model has better performance (lower MAE and RMSE) than other baseline and state-of-the-art models. Here, we use a statistical test which shows whether the difference of forecasting models are statistically significant. We use the Diebold Mariano (DM) test, as introduced in [42], which has become a well accepted hypothesis testing method.

Given $\mathbf{e}_{ts}^{m1}$ as the difference of actual flow and predicted flow in model m_1 at time stamp t

and sensor s, and an evaluation function $\mathsf{f}(.)$, such as MSE or MAE, a null hypothesis and an alternate hypothesis between two models m_1 and m_2 are defined as follows,

$$\mathsf{H}_0 : \mathbb{E}(\mathsf{f}(\mathbf{e}_{\mathsf{ts}}^{\mathsf{m}1})) = \mathbb{E}(\mathsf{f}(\mathbf{e}_{\mathsf{ts}}^{\mathsf{m}2})) \tag{5.7}$$

$$\mathsf{H}_\mathsf{a} : \mathbb{E}(\mathsf{f}(\mathbf{e}_{\mathsf{ts}}^{\mathsf{m}1})) \neq \mathbb{E}(\mathsf{f}(\mathbf{e}_{\mathsf{ts}}^{\mathsf{m}2})) \tag{5.8}$$

The null hypothesis indicates that the difference between expected prediction errors is zero. We can also represent the null hyposthesis with $\mathbf{d} = \mathbb{E}(\mathsf{f}(\mathbf{e}_{\mathsf{ts}}^{\mathsf{m}1}) - \mathsf{f}(\mathbf{e}_{\mathsf{ts}}^{\mathsf{m}2})) = 0$. An alternate hypothesis indicates that the performance of one model is significantly different than the other one. A statistical test evaluates these hypotheses. We can reject the null hypothesis if the p-value is lower than the significance level α (or if the DM test value is not in range of $[-Z_{\frac{\alpha}{2}}, Z_{\frac{\alpha}{2}}]$). If we reject the null hypothesis, then we conclude that the forecasting performance of the proposed model is significantly better than other models.

To apply the DM test on spatio-temporal data, we consider independent tests for spatial and temporal features. In some previous works, other researchers used independent DM tests for some specific time stamps, e.g. hours of a day [70], and independent DM tests for different locations and time horizons [56]. Such analysis shows that the accuracy of a forecasting model is significantly better than other models for specific temporal or spatial states. We follow such a procedure to test the statistical significance of forecasting models. We obtain the number of sensors, out of 380, for which the accuracy of the proposed model is statistically significant compared with the MLP and the LSTM models. We select MLP and LSTM models as the worst and best implemented neural networks. We compare these models with the best CNN-LSTM model (C-CNN-LSTM-DA), which is the best proposed model in this chapter. We perform a two-sided DM test with 1%, 5% and 10% significance levels. Table 5.8 shows that a large number of sensors have statistically significant lower MAE

and MSE. Some of the sensors have a very smooth traffic flows. Most of the models easily predicts their traffic flow data. In such sensors, the accuracy of the proposed model is almost the same as LSTM and it is not statistically significant. Also, another conclusion is that the MAE and MSE increases for longer time horizons, e.g. $\mathsf{h} = 12$ or 60-min prediction horizon. For a longer prediction horizon, the model differences are not statistically significant, which can be result of higher fluctuations in some of the sensors. However, based on the analysis in Section 5.8, the main difference of the performance of proposed model and other models is for peak hours. Hence, we examine the DM test only for peak hours. For 1% significance level, the number of sensors for which the proposed model has statistically significantly lower MAE and MSE values, are 366 and 358. However, the p-value for all of the sensors is less than 5%. Hence, for 5% significance level, the proposed model has statistically significant lower prediction error in all sensors.

Table 5.8: The number of sensors, out of 380, for which the proposed model has statistically significant lower MAE and MSE.

Models	Metric	α	15min	30min	45min	60min
MLP	MAE	1	380	380	380	380
		5	380	380	380	380
		10	380	380	380	380
	MSE	1	380	380	380	380
		5	380	380	380	380
		10	380	380	380	380
LSTM	MAE	1	380	358	308	290
		5	380	358	316	290
		10	380	358	331	305
	MSE	1	371	313	281	245
		5	376	323	292	268
		10	380	335	317	290

5.6 Conclusion and future work

This chapter proposes a new deep learning framework for spatio-temporal forecasting problems. The full results of this study were first published in [16]. We illustrate the application

of this framework on traffic flow data. The proposed framework consists of several components, each of which are carefully designed based on spatio-temporal patterns. In Section 5.2 and 5.3, we described the main spatio-temporal patterns in traffic data. We thoroughly analyze the performance of the proposed model compared with other implemented models. Table 5.3 shows the comparison of the baseline, state-of-the-art neural network models, and the proposed CNN-LSTM models. The C-CNN-LSTM models have the lowest prediction error. In Table 5.6, we evaluate the prediction of time series residuals. In off-peak hours, the performance of the proposed model is the same as the LSTM model. However, the proposed model has better performance in peak hours. We use a large number of traffic sensors in our experimental results. This large data set allows us to better evaluate models over various types of spatial and temporal features. In Section 5.5.9 and 5.5.10, we analyze prediction errors over various spatial and temporal features. Also, the Diebold-Mariano test shows that the proposed model has statistically significant lower prediction error. Moreover, we show that the pretrained autoencoder decoder, as the last component of C-CNN-LSTM-DA, reduces prediction error where there are missing data, Fig. 5.11.

While deep learning models have a great success in various domains, such as computer vision or natural language processing, there are few studies that develop new deep learning models for spatio-temporal problems. This study demonstrates the effectiveness of designing new neural network architectures considering specific properties of spatio-temporal problems. While the focus of our work is on the forecasting problem, related models can be developed for other spatio-temporal problems, such as anomaly detection, missing data imputation, time series clustering, and time series classification problems. In addition, we focus on traffic flow data which has non-linear correlation in neighboring spatial and temporal features. Other types of spatio-temporal problems have their unique spatial and temporal patterns. The proposed model can be applied on various spatio-temporal applications, but with proper attention given to their unique spatial, short-term and long-term patterns.

Bibliography

[1] California. pems, `http://pems.dot.ca.gov/`, 2017.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[3] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

[4] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[5] M. Ahmed and A. N. Mahmood. Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection. *Annals of Data Science*, 2(1):111–130, 2015.

[6] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha. Efficient machine learning for big data: A review. *Big Data Research*, 2(3):87–93, 2015.

[7] B. Anbaroglu, B. Heydecker, and T. Cheng. Spatio-temporal clustering for non-recurrent traffic congestion detection on urban road networks. *Transportation Research Part C: Emerging Technologies*, 48:47–65, 2014.

[8] M. Y. Ansari, A. Ahmad, S. S. Khan, G. Bhushan, et al. Spatiotemporal clustering: a review. *Artificial Intelligence Review*, pages 1–43, 2019.

[9] C. F. Ansley and R. Kohn. On the estimation of arima models with missing values. In *Time series analysis of irregularly observed data*, pages 9–37. Springer, 1984.

[10] R. Asadi and M. Ghatee. A rule-based decision support system in intelligent hazmat transportation system. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2756–2764, 2015.

[11] R. Asadi and S. S. Kia. Cycle flow formulation of optimal network flow problems and respective distributed solutions. *IEEE/CAA Journal of Automatica Sinica*, 6(5):1251–1260, 2019.

[12] R. Asadi, S. S. Kia, and A. Regan. Cycle basis distributed admm solution for optimal network flow problem over biconnected graphs. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 717–724. IEEE, 2016.

[13] R. Asadi and A. Regan. A convolution recurrent autoencoder for spatio-temporal missing data imputation. *arXiv preprint arXiv:1904.12413*, 2019.

[14] R. Asadi and A. Regan. A convolutional recurrent autoencoder for spatio-temporal missing data imputation. *International Conference on Artificial Intelligence (ICAI 2019)*, 2019.

[15] R. Asadi and A. Regan. Spatio-temporal clustering of traffic data with deep embedded clustering. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Prediction of Human Mobility*, pages 45–52, 2019.

[16] R. Asadi and A. C. Regan. A spatio-temporal decomposition based deep neural network for time series forecasting. *Applied Soft Computing*, 87:105963, 2020.

[17] G. Atluri, A. Karpatne, and V. Kumar. Spatio-temporal data mining: A survey of problems and methods. *ACM Computing Surveys (CSUR)*, 51(4):83, 2018.

[18] M. Azimi and Y. Zhang. Categorizing freeway flow conditions by using clustering methods. *Transportation Research Record*, 2173(1):105–114, 2010.

[19] Q. Ba, K. Savla, and G. Como. Distributed optimal equilibrium selection for traffic flow over networks. In *IEEE Conference on Decision and Control*, 2015.

[20] B. Bae, H. Kim, H. Lim, Y. Liu, L. D. Han, and P. B. Freeze. Missing data imputation for traffic flow speed using spatio-temporal cokriging. *Transportation Research Part C: Emerging Technologies*, 88:124–139, 2018.

[21] J. Barros, M. Araujo, and R. J. Rossetti. Short-term real-time traffic prediction methods: A survey. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 132–139. IEEE, 2015.

[22] J. Béjar, S. Álvarez, D. García, I. Gómez, L. Oliva, A. Tejeda, and J. Vázquez-Salceda. Discovery of spatio-temporal patterns from location-based social networks. *Journal of Experimental & Theoretical Artificial Intelligence*, 28(1-2):313–329, 2016.

[23] D. P. Bertsekas. *Network optimization: continuous and discrete models*. Citeseer, 1998.

[24] R. J. Bessa, A. Trindade, and V. Miranda. Spatial-temporal solar power forecasting for smart grids. *IEEE Transactions on Industrial Informatics*, 11(1):232–241, 2015.

[25] E. Bolshinsky and R. Friedman. Traffic flow forecast survey. Technical report, Computer Science Department, Technion, 2012.

[26] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[27] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[28] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[29] L. Caggiani, M. Ottomanelli, R. Camporeale, and M. Binetti. Spatio-temporal clustering and forecasting method for free-floating bike sharing systems. In *International Conference on Systems Science*, pages 244–254. Springer, 2016.

[30] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li. Brits: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems*, pages 6776–6786, 2018.

[31] H. B. Celikoglu and M. A. Silgu. Extension of traffic flow pattern dynamic classification by a macroscopic model using multivariate clustering. *Transportation Science*, 50(3):966–981, 2016.

[32] X. Cheng, R. Zhang, J. Zhou, and W. Xu. Deeptransport: Learning spatial-temporal dependency for traffic condition forecasting. *arXiv preprint arXiv:1709.09585*, 2017.

[33] F. Chollet et al. Keras, 2015.

[34] M. Y. Choong, L. Angeline, R. K. Y. Chin, K. B. Yeo, and K. T. K. Teo. Vehicle trajectory clustering for traffic intersection surveillance. In *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–4. IEEE, 2016.

[35] J. Chow. *Informed Urban Transport Systems: Classic and Emerging Mobility Methods toward Smart Cities*. Elsevier, 2018.

[36] H. Chunchun, L. Nianxue, Y. Xiaohong, and S. Wenzhong. Traffic flow data mining and evaluation based on fuzzy clustering techniques. *International Journal of Fuzzy Systems*, 13(4), 2011.

[37] A. F. Costa, M. S. Santos, J. P. Soares, and P. H. Abreu. Missing data imputation via denoising autoencoders: The untold story. In *International Symposium on Intelligent Data Analysis*, pages 87–98. Springer, 2018.

[38] C. F. Daganzo. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, 28(4):269–287, 1994.

[39] C. De-Wang. Classification of traffic flow situation of urban freeways based on fuzzy clustering [j]. *Communication and Transportati0n Systems Engineering and Information*, 1, 2005.

[40] S. Deng, S. Jia, and J. Chen. Exploring spatial–temporal relations via deep convolutional neural networks for traffic flow prediction with incomplete data. *Applied Soft Computing, 2018*, 2018.

[41] A. Dharwadker and S. Pirzada. *Graph Theory*. CreateSpace Independent Publishing Platform, 2011.

[42] F. X. Diebold and R. S. Mariano. Comparing predictive accuracy. *Journal of Business & economic statistics*, 20(1):134–144, 2002.

[43] Y. Duan, Y. Lv, Y.-L. Liu, and F.-Y. Wang. An efficient realization of deep learning for traffic data imputation. *Transportation research part C: emerging technologies*, 72:168–181, 2016.

[44] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[45] D. C. Gazis and C. H. Knapp. On-line estimation of traffic densities from time-series of flow and speed data. *Transportation Science*, 5(3):283–301, 1971.

[46] L. Gondara and K. Wang. Mida: Multiple imputation using denoising autoencoders. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 260–272. Springer, 2018.

[47] S. Guo, Y. Lin, S. Li, Z. Chen, and H. Wan. Deep spatial-temporal 3d convolutional neural networks for traffic data forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[48] X. Guo, L. Gao, X. Liu, and J. Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, pages 1753–1759, 2017.

[49] L. Gupta, D. L. Molfese, R. Tammana, and P. G. Simos. Nonlinear alignment and averaging for estimating the evoked potential. *IEEE Transactions on Biomedical Engineering*, 43(4):348–356, 1996.

[50] S. Hajiseyedjavadi, Y.-R. Lin, and K. Pelechrinis. Discovering functionality of urban regions by learning low-dimensional representations of a spatial multiplex network. In *Proceedings of the Third Mining Urban Data Workshop (MUD 2018)*, 2018.

[51] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[52] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[53] W. Huang, G. Song, H. Hong, and K. Xie. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2191–2201, 2014.

[54] X. Huang, Y. Ye, L. Xiong, R. Y. Lau, N. Jiang, and S. Wang. Time series k-means: A new k-means type smooth subspace clustering for time series data. *Information Sciences*, 367:1–13, 2016.

[55] Y. Jia, C. Zhou, and M. Motani. Spatio-temporal autoencoder for feature learning in patient data with missing observations. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 886–890. IEEE, 2017.

[56] P. Jiang and Z. Liu. Variable weights combined model based on multi-objective optimization for short-term wind speed forecasting. *Applied Soft Computing*, page 105587, 2019.

[57] Y. Kamarianakis and P. Prastacos. Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches. *Transportation Research Record*, 1857(1):74–84, 2003.

[58] M. G. Karlaftis and E. I. Vlahogianni. Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation Research Part C: Emerging Technologies*, 19(3):387–399, 2011.

[59] B. H. Kim and R. Baldick. A comparison of distributed optimal power flow algorithms. *IEEE Transactions on Power Systems*, 15(2):599–604, 2000.

[60] Y. Kim, P. Wang, and L. Mihaylova. Structural recurrent neural network for traffic speed prediction. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5207–5211. IEEE, 2019.

[61] S. Kisilevich, F. Mansmann, M. Nanni, and S. Rinzivillo. Spatio-temporal clustering. In *Data mining and knowledge discovery handbook*, pages 855–874. Springer, 2009.

[62] T. Koide, H. Kubo, and H. Watanabe. A study on the tie-set graph theory and network flow optimization problems. *International Journal of Circuit Theory and Applications*, 32(6):447–470, 2004.

[63] T. Koide and H. Watanabe. A theory of tie-set graph and tie-set path-a graph theoretical study on robust network system. In *Circuits and Systems, 2000. IEEE APCCAS 2000. The 2000 IEEE Asia-Pacific Conference on*, pages 227–230. IEEE, 2000.

[64] V. Kolmogorov and A. Shioura. New algorithms for the dual of the convex cost network flow problem with application to computer vision. *Mathematical Programming*, 2007.

[65] M. Konkol. Fuzzy agglomerative clustering. In *International Conference on Artificial Intelligence and Soft Computing*, pages 207–217. Springer, 2015.

[66] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[67] W. C. Ku, G. R. Jagadeesh, A. Prakash, and T. Srikanthan. A clustering-based approach for data-driven imputation of missing traffic data. In *2016 IEEE Forum on Integrated and Sustainable Transportation Systems (FISTS)*, pages 1–6. IEEE, 2016.

[68] S. V. Kumar and L. Vanajakshi. Short-term traffic flow prediction using seasonal arima model with limited input data. *European Transport Research Review*, 7(3):21, 2015.

[69] T. Kuremoto, S. Kimura, K. Kobayashi, and M. Obayashi. Time series forecasting using a deep belief network with restricted boltzmann machines. *Neurocomputing*, 137:47–56, 2014.

[70] J. Lago, F. De Ridder, and B. De Schutter. Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms. *Applied Energy*, 221:386–405, 2018.

[71] I. Laña, I. I. Olabarrieta, M. Vélez, and J. Del Ser. On the imputation of missing data for road traffic forecasting: New insights and novel techniques. *Transportation research part C: emerging technologies*, 90:18–33, 2018.

[72] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. *arXiv preprint arXiv:1112.6209*, 2011.

[73] L. Li, J. Zhang, Y. Wang, and B. Ran. Missing value imputation for traffic-related time series data based on a multi-view learning method. *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[74] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[75] Q. Liu, B. Wang, and Y. Zhu. Short-term traffic speed forecasting based on attention convolutional neural network for arterials. *Computer-Aided Civil and Infrastructure Engineering*, 33(11):999–1016, 2018.

[76] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1010–1018. ACM, 2011.

[77] Z. Liu, S. Sharma, and S. Datla. Imputation of missing traffic data during holiday periods. *Transportation Planning and Technology*, 31(5):525–544, 2008.

[78] I. Lobel and A. Ozdaglar. Distributed subgradient methods for convex optimization over random networks. *Automatic Control, IEEE Transactions on*, 56(6):1291–1306, 2011.

[79] Q. Lu, F. Chen, and K. Hancock. On path anomaly detection in a large transportation network. *Computers, Environment and Urban Systems*, 33(6):448–462, 2009.

[80] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2014.

[81] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

[82] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, 17(4):818, 2017.

[83] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54:187–197, 2015.

[84] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[85] S. Mac Lane. *A combinatorial condition for planar graphs*. Seminarium Matemat., 1936.

[86] K. Ø. Mikalsen, F. M. Bianchi, C. Soguero-Ruiz, and R. Jenssen. Time series cluster kernel for learning similarities between multivariate time series with missing data. *Pattern Recognition*, 76:569–581, 2018.

[87] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[88] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Puschel. Distributed optimization with local domains: Applications in mpc and network flows. *Automatic Control, IEEE Transactions on*, 60(7):2004–2009, 2015.

[89] L. Muñoz, X. Sun, R. Horowitz, and L. Alvarez. Traffic density estimation with the cell transmission model. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 5, pages 3750–3755. IEEE, 2003.

[90] A. Muralidharan and R. Horowitz. Imputation of ramp flow data for freeway traffic simulation. *Transportation Research Record*, 2099(1):58–64, 2009.

[91] A. M. Nagy and V. Simon. Survey on traffic prediction in smart cities. *Pervasive and Mobile Computing*, 50:148–163, 2018.

[92] K. Nakayama, C. Zhao, L. F. Bic, M. B. Dillencourt, and J. Brouwer. Distributed power flow loss minimization control for future grid. *International Journal of Circuit Theory and Applications*, 43(9):1209–1225, 2015.

[93] R. P. D. Nath, H.-J. Lee, N. K. Chowdhury, and J.-W. Chang. Modified k-means clustering for travel time prediction based on historical traffic data. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 511–521. Springer, 2010.

[94] A. Nedić and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *Automatic Control, IEEE Transactions on*, 54(1):48–61, 2009.

[95] V. Niennattrakul and C. A. Ratanamahatana. Shape averaging under time warping. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*, volume 2, pages 626–629. IEEE, 2009.

[96] J. Paparrizos and L. Gravano. Fast and accurate time-series clustering. *ACM Transactions on Database Systems (TODS)*, 42(2):8, 2017.

[97] B. Park, C. J. Messer, and T. Urbanik. Short-term freeway traffic volume forecasting using radial basis function neural network. *Transportation Research Record*, 1651(1):39–47, 1998.

[98] Q. Peng and S. H. Low. Distributed algorithm for optimal power flow on a radial network. In *53rd IEEE Conference on Decision and Control*, pages 167–172. IEEE, 2014.

[99] F. Petitjean and P. Gançarski. Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment. *Theoretical Computer Science*, 414(1):76–91, 2012.

[100] N. G. Polson and V. O. Sokolov. Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 79:1–17, 2017.

[101] X. Qiu, Y. Ren, P. N. Suganthan, and G. A. Amaratunga. Empirical mode decomposition based ensemble deep learning for load demand time series forecasting. *Applied Soft Computing*, 54:246–255, 2017.

[102] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga. Ensemble deep learning for regression and time series forecasting. In *Computational Intelligence in Ensemble Learning (CIEL), 2014 IEEE Symposium on*, pages 1–6. IEEE, 2014.

[103] L. Qu, L. Li, Y. Zhang, and J. Hu. Ppca-based missing data imputation for traffic flow volume: A systematical approach. *IEEE Transactions on intelligent transportation systems*, 10(3):512–522, 2009.

[104] L. Qu, Y. Zhang, J. Hu, L. Jia, and L. Li. A bpca based missing value imputing method for traffic flow volume data. In *2008 IEEE Intelligent Vehicles Symposium*, pages 985–990. IEEE, 2008.

[105] B. Ran, H. Tan, Y. Wu, and P. J. Jin. Tensor based missing traffic data completion with spatial–temporal correlation. *Physica A: Statistical Mechanics and its Applications*, 446:54–63, 2016.

[106] F. Rempe, G. Huber, and K. Bogenberger. Spatio-temporal congestion patterns in urban traffic networks. *Transportation Research Procedia*, 15:513–524, 2016.

[107] S. Rezaei, K. Kim, and E. Bozorgzadeh. Scalable multi-queue data transfer scheme for fpga-based multi-accelerators. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*, pages 374–380. IEEE, 2018.

[108] A. W. Sadek, G. Spring, and B. L. Smith. Toward more effective transportation applications of computational intelligence paradigms. *Transportation research record*, 1836(1):57–63, 2003.

[109] H. Sak, A. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014.

[110] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[111] J. L. Schafer and M. K. Olsen. Multiple imputation for multivariate missing-data problems: A data analyst's perspective. *Multivariate behavioral research*, 33(4):545–571, 1998.

[112] T. Seo, A. M. Bayen, T. Kusakabe, and Y. Asakura. Traffic state estimation on highway: A comprehensive survey. *Annual reviews in control*, 43:128–151, 2017.

[113] X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. In *Advances in neural information processing systems*, pages 5617–5627, 2017.

[114] N. Z. Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

[115] M. A. Silgu and H. B. Celikoglu. Clustering traffic flow patterns by fuzzy c-means method: some preliminary findings. In *International Conference on Computer Aided Systems Theory*, pages 756–764. Springer, 2015.

[116] B. L. Smith and M. J. Demetsky. Short-term traffic flow prediction models-a comparison of neural network and nonparametric regression approaches. In *Systems, Man, and Cybernetics, 1994. Humans, Information and Technology., 1994 IEEE International Conference on*, volume 2, pages 1706–1709. IEEE, 1994.

[117] S. Soheily-Khah, A. Douzal-Chouakria, and E. Gaussier. Generalized k-means-based clustering for temporal data under weighted and kernel time warp. *Pattern Recognition Letters*, 75:63–69, 2016.

[118] J. Tang, G. Zhang, Y. Wang, H. Wang, and F. Liu. A hybrid approach to integrate fuzzy c-means based imputation method with genetic algorithm for missing traffic volume data estimation. *Transportation Research Part C: Emerging Technologies*, 51:29–40, 2015.

[119] A. Tascikaraoglu. Evaluation of spatio-temporal forecasting methods in various smart city applications. *Renewable and Sustainable Energy Reviews*, 82:424–435, 2018.

[120] A. Tascikaraoglu and B. M. Sanandaji. Short-term residential electric load forecasting: A compressive spatio-temporal approach. *Energy and Buildings*, 111:380–392, 2016.

[121] J. Tastu, P. Pinson, E. Kotwa, H. Madsen, and H. A. Nielsen. Spatio-temporal analysis and modeling of short-term wind power forecast errors. *Wind Energy*, 14(1):43–60, 2011.

[122] D. A. Thomas and J. F. Weng. Minimum cost flow-dependent communication networks. *Networks*, 48(1):39–46, 2006.

[123] Y. Tian, K. Zhang, J. Li, X. Lin, and B. Yang. Lstm-based traffic flow prediction with missing data. *Neurocomputing*, 318:297–305, 2018.

[124] M. Torabzadehkashi, S. Rezaei, A. Heydarigorji, H. Bobarshad, V. Alves, and N. Bagherzadeh. Catalina: In-storage processing acceleration for scalable big data analytics. In *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 430–437. IEEE, 2019.

[125] P. Tzirakis, M. A. Nicolaou, B. Schuller, and S. Zafeiriou. Time-series clustering with jointly learning deep representations, clusters and temporal boundaries.

[126] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[127] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408, 2010.

[128] C. Wang, X. Li, X. Zhou, A. Wang, and N. Nedjah. Soft computing in big data intelligent transportation systems. *Applied Soft Computing*, 38:1099–1108, 2016.

[129] J. Wang, W. Deng, and Y. Guo. New bayesian combination method for short-term traffic flow forecasting. *Transportation Research Part C: Emerging Technologies*, 43:79–94, 2014.

[130] L. Wang, Z. Wang, H. Qu, and S. Liu. Optimal forecast combination based on neural networks for time series forecasting. *Applied Soft Computing*, 66:1–17, 2018.

[131] M. Wang, A. Ailamaki, and C. Faloutsos. Capturing the spatio-temporal behavior of real traffic data. *Performance Evaluation*, 49(1-4):147–163, 2002.

[132] S. Wang, J. Cao, and P. S. Yu. Deep learning for spatio-temporal data mining: A survey. *arXiv preprint arXiv:1906.04928*, 2019.

[133] Y. Wu, A. Ribeiro, and G. B. Giannakis. Robust routing in wireless multi-hop networks. In *Information Sciences and Systems, 2007. CISS'07. 41st Annual Conference on*, pages 637–642. IEEE, 2007.

[134] Y. Wu, H. Tan, L. Qin, B. Ran, and Z. Jiang. A hybrid deep learning based traffic flow prediction method and its understanding. *Transportation Research Part C: Emerging Technologies*, 90:166–180, 2018.

[135] L. Xiao, M. Johansson, and S. P. Boyd. Simultaneous routing and resource allocation via dual decomposition. *Communications, IEEE Transactions on*, 52(7):1136–1144, 2004.

[136] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[137] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[138] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3861–3870. JMLR. org, 2017.

[139] S. Yi, J. Ju, M.-K. Yoon, and J. Choi. Grouped convolutional neural networks for multivariate time series. *arXiv preprint arXiv:1703.09938*, 2017.

[140] G. Yu, J. Hu, C. Zhang, L. Zhuang, and J. Song. Short-term traffic flow forecasting based on markov chain model. In *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, pages 208–212. IEEE, 2003.

[141] H.-F. Yu, N. Rao, and I. S. Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. In *Advances in neural information processing systems*, pages 847–855, 2016.

[142] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie. Accelerated dual descent for network flow optimization. *Automatic Control, IEEE Transactions on*, 59(4):905–920, 2014.

[143] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[144] W. Zhang, Y. Yu, Y. Qi, F. Shu, and Y. Wang. Short-term traffic flow prediction based on spatio-temporal analysis and cnn deep learning. *Transportmetrica A: Transport Science*, 15(2):1688–1711, 2019.

[145] Y.-F. Zhang, P. Thorburn, W. Xiang, and P. Fitch. Ssim-a deep learning approach for recovering missing time series sensor data. *IEEE Internet of Things Journal*, 2019.

[146] Z. Zhang, M. Li, X. Lin, Y. Wang, and F. He. Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies. *Transportation Research Part C: Emerging Technologies*, 105:297–322, 2019.

[147] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu. Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2):68–75, 2017.

[148] X. Zheng, W. Chen, P. Wang, D. Shen, S. Chen, X. Wang, Q. Zhang, and L. Yang. Big data for social transportation. *IEEE Transactions on Intelligent Transportation Systems*, 17(3):620–630, 2016.

[149] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.

[150] T. Zhou, G. Han, X. Xu, Z. Lin, C. Han, Y. Huang, and J. Qin. δ-agree adaboost stacked autoencoder for short-term traffic flow forecasting. *Neurocomputing*, 247:31–38, 2017.

[151] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang. Big data analytics in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 20(1):383–398, 2018.

[152] Y. Zhuang, R. Ke, and Y. Wang. Innovative method for traffic data imputation based on convolutional neural network. *IET Intelligent Transport Systems*, 2018.